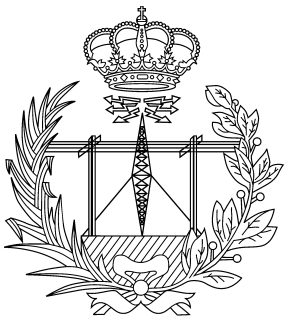
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS
INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA



Trabajo Fin de Grado

ANÁLISIS DE RENDIMIENTO DE REDES VRAN EN ESCENARIOS CON LIMITACIÓN DE RECURSOS DE VIRTUALIZACIÓN

(Performance analysis of vRAN architectures over resource-limited scenarios)

Para acceder al Título de

Graduado en
Ingeniería de Tecnologías de Telecomunicación

Autor: Daniel Martínez Rica

Septiembre- 2021

UNIVERSIDAD DE CANTABRIA

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Daniel Martínez Rica
Director del TFG: Ramón Agüero Calvo, Luis Francisco Díez Fernández
Título: "ANÁLISIS DE RENDIMIENTO DE REDES VRAN EN ESCENARIOS CON LIMITACIÓN DE RECURSOS DE VIRTUALIZACIÓN"
Title: "Performance analysis of vRAN architectures over resource-limited scenarios"

Presentado a examen el día: 20 de septiembre de 2021

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del tribunal:
Presidente (Apellidos, Nombre): Pedro Corcuera Miro-Quesada
Secretario (Apellidos, Nombre): José Ángel Irastorza Teja
Vocal (Apellidos, Nombre): Ramón Agüero Calvo

Este Tribunal ha resulto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFG
(solo si es distinto del Secretario)

V^{o} B^{o} del Subdirector

Trabajo Fin de Grado N^{o}
(a asignar por Secretaría)

Índice

Índice de figuras

Índice de tablas

Índice de acrónimos

3GPP 3rd Generation Partnership Project.

BBU Baseband Unit.

BP Backpressure.

BS estación base.

C-RAN Centralized Radio Access Networks.

CU Central Unit.

D-RAN Fully-Decentralized Radio Access Networks.

DU Distributed Unit.

ECDF Función de Distribución Acumulativa Empírica.

eMBB enhanced mobile broadband.

GOPS Gigaoperaciones por segundo.

mMTC massive machine-type communications.

RR Round Robin.

RRH Remote Radio Head Unit.

RU Remote Unit.

URLLC ultra-reliable low latency communications.

WFQ Weighted Fair Queuing.

Chapter 1

Introducción

La demanda de los consumidores está configurando el desarrollo de los servicios de banda ancha móvil. El aumento de tráfico, el incremento del número de dispositivos y servicios y la demanda de una mejor experiencia de usuario requieren de nuevas soluciones. Se espera que la quinta generación de tecnologías móviles cumpla con estos requisitos. Según el *Cisco Annual Internet Report* [1], la red 5G admitirá más del 10% de las conexiones móviles del mundo para el año 2023.

Para ello, en [2], la ITU-R definió los requisitos para las redes 5G, IMT-2020, que contemplan tres casos de uso principales para las redes móviles: *enhanced Mobile BroadBand (eMBB), massive Machine-Type Communications (mMTC) y Ultra Reliable and Low Latency Communications (URLLC).*

El primer caso de uso, eMBB, se centra en la utilización de la red por personas y permitirá tener banda de ancha mejorada en entornos interiores y exteriores, así como servicios de realidad virtual y aumentada. El segundo, mMTC, será clave para el desarrollo del Internet de las Cosas (IoT) así como de ciudades, hogares y servicios inteligentes. Por último, el URLLC busca una baja latencia y se centrará en vehículos autónomos, servicios de telesalud, automatización industrial y redes eléctricas inteligentes.

Con estos ambiciosos objetivos, la redes 5G se enfrentan a retos importantes. El aumento de la capacidad y de las velocidades de datos prometidas por las redes de quinta generación requiere más espectro y tecnologías capaces de utilizar estos recursos de manera mucho más eficiente.

Una de las soluciones principales es la *Cloud Radio Access Network (C-RAN)* en la que se aprovechan las ventajas de la computación en la nube para obtener mayores capacidades y menores retardos. En concreto, en las arquitecturas C-RAN se virtualizan y centralizan los protocolos y funciones de las estaciones base, que tradicionalmente se encontraban junto a las antenas. Esta centralización permite una mejor cooperación entre los elementos de acceso, lo que redunda en un uso más eficiente de los recursos. Sin embargo, esta centralización puede no resultar asumible en algunos casos, debido a las altas capacidades que se impondrían en la red *fronthaul* que comunica los puntos donde se instalan las antenas y las centrales de procesado donde se encuentran las unidades virtualizadas [3].

Para aliviar los requisitos de la red *fronthaul* una de las ideas que toma más fuerza es el *functional split*, en el que se seleccionan diferentes grados de centralización en función de parámetros como el estado de los enlaces, requisitos de retardo o el tráfico de usuario.

La selección del nivel de *split* puede fijarse a priori. Sin embargo, esto puede no ser el método más

eficiente, puesto que no se tiene en cuenta la variabilidad del tráfico de usuario, entre otras variables. Por esta razón, se han propuesto soluciones para llevar a cabo la selección de *split* de manera dinámica.

Para ello se han propuesto diferentes soluciones de selección de *split*, normalmente centradas en la reducción del retardo. A su vez, la selección debe tener en cuenta los recursos de procesado disponibles en los centros de procesado, que son compartidos por las estaciones base virtualizadas.

En este trabajo se va a analizar la posibilidad de utilizar teoría de control para llevar a cabo la selección de *split* en entornos con limitación de recursos en los centros de procesado. Concretamente se analizará el algoritmo de *Backpressure* que ha sido ampliamente usado en encaminamiento de redes y que garantiza la estabilidad del sistema, con retardo reducido. En concreto, los objetivos de este trabajo son los siguientes:

- Implementación, mediante sistema de colas, de los nodos distribuidos y centralizados que componen la arquitectura 5G vRAN.
- Evaluación de varios algoritmos de *scheduling* para la selección dinámica de *splits*, y cuyo comportamiento sirva de referencia para evaluar el rendimiento de *Backpressure*.
- Análisis de la selección de *splits* en escenarios realistas donde los recursos computacionales compartidos entre las estaciones base son un factor limitante.

Este trabajo sigue la estructura descrita a continuación. En el **Capítulo 2** se describe la arquitectura 5G vRAN sobre la que se desarrollan los escenarios a analizar y se presentan los algoritmos de *scheduling* que se utilizan en la selección de *functional splits*. Después, en el **Capítulo 3** se detalla la implementación en Matlab del sistema y los algoritmos descritos en el **Capítulo 2**, además de las métricas con las que se evalúan los resultados. A continuación, en el **Capítulo 4** se definen los escenarios en los que se pone a prueba el rendimiento de *Backpressure* comparando su comportamiento con el resto de algoritmos. Finalmente, en el **Capítulo 5** se lleva a cabo la conclusión del trabajo.

Chapter 2

Estado del arte

En esta sección se van a presentar los aspectos y conceptos utilizados para llevar a cabo el trabajo. En primer lugar se describirán las arquitecturas 5G vRAN, sobre las que se desarrollarán los escenarios que se van a analizar. Seguidamente, se va a presentar un conjunto de algoritmos cuyo rendimiento se analizará en secciones posteriores.

2.1 Arquitecturas *virtual Radio Access Network* (vRAN)

La llegada de la quinta generación de redes móviles promete mejorar considerablemente las prestaciones ya existentes, considerando tres casos de uso: *enhanced mobile broadband (eMBB)*, *ultra-reliable low latency communications (URLLC)* y *massive machine-type communications (mMTC)* [4]. Estos objetivos requieren una utilización eficiente del canal y de los recursos computacionales disponibles. Para ello, es necesario una renovación de la infraestructura de la red que ha sido constante desde las primeras generaciones de redes móviles.

En las redes 3G se introdujo la división funcional de la estación base (BS) en dos unidades, la *Remote Radio Head Unit (RRH)* y la *Baseband Unit (BBU)* [5]. En este contexto el procesado en banda base y los elementos radio se ubican en un mismo punto conformando la denominada *Fully-Decentralized Radio Access Networks (D-RAN)* [6] que cambiaría con la llegada de la cuarta generación de redes móviles.

Las redes 4G introdujeron el concepto de *Centralized Radio Access Networks (C-RAN)*, donde las BBUs son centralizadas y virtualizadas haciendo uso de un conjunto de procesadores de propósito general denominados *BBU-Pools* [5]. Esta centralización permite cooperación más estrecha entre las BSs mediante las interacción de sus BBUs que se encuentran físicamente juntas. A su vez, este mayor grado de cooperación posibilita explotar técnicas avanzadas que incrementen la utilización más eficiente de los recursos de comunicación. Por otro lado, la centralización también reduce costes de operación y despliegue, otorga una mayor flexibilidad y adaptabilidad en la red así como una reducción del consumo energético [7].

Sin embargo, C-RAN presenta un gran inconveniente: requiere una red *fronthaul* con alta capacidad y baja latencia para soportar el tráfico que conecta los centros de procesado donde se encuentran las BBUs, con los puntos de acceso remoto [8].

Para solucionar este problema la organización *3rd Generation Partnership Project (3GPP)* propuso

una nueva arquitectura parcialmente centralizada [9] en la que la *BBU-pool* es dividida en dos partes: *Central Unit (CU)* que se instancia en servidores centralizados y *Distributed Unit (DU)*. Por su parte la RRH pasa a llamarse *Remote Unit (RU)*. Esta nueva arquitectura de 3 capas se adapta mejor que la C-RAN a los requisitos heterogéneos del 5G, recibiendo ahora el nombre de 5G-RAN o V-RAN.

Así una parte de las tareas tradicionales de la BBU se procesan de manera centralizada en la CU mientras que la otra parte de las tareas restantes se procesa de manera distribuida en la DU, dando lugar a los *functional splits*. En [9] el 3GPP introduce ocho opciones de split con el fin de adaptar la red a la quinta generación de redes móviles. Estas ocho opciones son las que se muestran en la figura 2.1 y son las siguientes:

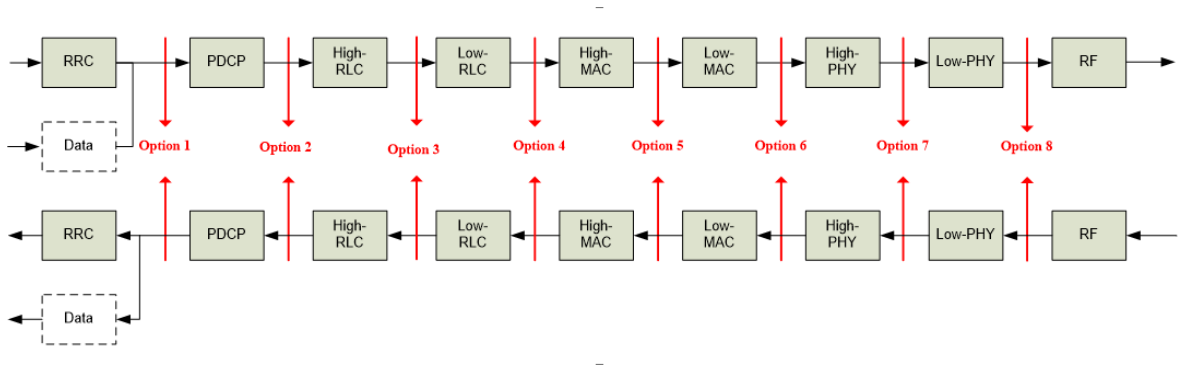


Figura 2.1: Opciones de división funcional entre CU y DU

- Opción 1: RRC está en la CU. PDCP, RLC, MAC, capa física y RF están en la DU.
- Opción 2: RRC y PDCP están en la CU. RLC, MAC, capa física y RF están en la DU.
- Opción 3: RRC, PDCP y High-RLC (división parcial de RLC) están en la CU. Low-RLC, MAC capa física y RF están en la DU.
- Opción 4: RRC, PDCP y RLC están en la CU. MAC, capa física y RF están en la DU.
- Opción 5: RRC, PDCP, RLC y High-MAC están en la CU. Low-MAC, capa física y RF están en la DU.
- Opción 6: RRC, PDCP, RLC y MAC están en la CU. Capa física y RF están en la DU.
- Opción 7: Low-PHY y Rf en la DU. El resto de capas están en la CU.
- Opción 8: Únicamente RF está en la DU mientras que el resto de capas están en la CU.

Con esta nueva arquitectura la red de *fronthaul* se divide en dos segmentos: el que comunica las CUs y DUs (también llamado *midhaul*), y el que conecta a las DUs con sus RUs correspondientes. Como se ha mencionado, las funciones de bajo nivel, que son las más exigentes en cuanto a ancho de banda y latencia, se implementan en la DU, mientras que la CU alberga las capas altas del *stack* de la BBU. Teniendo en cuenta que las DU se ubicarán en puntos cercanos a las RU, esta solución en tres niveles reduce parcialmente los requisitos en la red *fronthaul*, al menos en el segmento entre las CUs y DUs.

El concepto de *functional split* permite solucionar parcialmente el problema de la red *fronthaul* pero presenta a su vez un nuevo reto: elegir la opción de *functional split* adecuada. Debido al crecimiento exponencial de la demanda de tráfico móvil, fijar a priori la opción de *functional split* no es la solución óptima [8].

Una selección dinámica de los *splits* que tenga en cuenta la variablidad del tráfico de usuario permite aprovechar de manera eficiente los recursos de la red *fronthaul*, proveer mejor servicio a los usuarios y aprovechar las ventajas tanto de la arquitectura D-RAN como de la C-RAN [6].

La selección dinámica de split es un área muy prometedor que aún se encuentra en fase de investigación. Para llevar a cabo esta tarea, en este trabajo se analizan tres algoritmos de *scheduling* cuyas características se muestran en el siguiente apartado.

2.2 Algoritmos implementados

En este trabajo se describen tres algoritmos de scheduling para la selección dinámica de splits. Los dos primeros (*Round Robin (RR)* y *Weighted Fair Queuing (WFQ)*) servirán de base para analizar el rendimiento del tercer algoritmo implementado *Backpressure (BP)* basado en teoría de Lyupanov y que, a diferencia de los anteriores, ofrece la selección dinámica de splits en tiempo real.

Antes de explicar los algoritmos, merece la pena indicar que en todos los casos se asume un sistema de tiempo ranurado. De esta manera, en cada ranura o *slot*, el nivel de *split* se escogerá en función del algoritmo utilizado.

2.2.1 Round Robin (RR)

El primer algoritmo de scheduling que se ha implementado para el correcto funcionamiento del sistema es el *Round Robin*. Su comportamiento es el más básico de todos: asigna la misma probabilidad a cada split funcional de manera que en cada slot temporal todas las opciones tienen la misma probabilidad de ser escogidos. Por ejemplo, en el caso de la arquitectura 5G-RAN con ocho splits, cada opción tiene la misma probabilidad de 1/8 de ser seleccionado tal y como se puede apreciar en la figura 2.2.

Split	1	2	3	4	5	6	7	8
Pr	0,125	0,125	0,125	0,125	0,125	0,125	0,125	0,125

Figura 2.2: Selección de *splits* de acuerdo a *Round Robin*

Se trata del algoritmo más básico pero también del menos eficiente. Como se demostrará más adelante, aunque la carga de tráfico se reparte de igual manera entre todas las opciones, esta asignación no tiene en cuenta las variaciones en cada *slot* de tiempo. Esto hace que no atienda a criterios como las capacidades de los enlaces o el tráfico de usuario, provocando alto retardo e inestabilidad en las colas.

2.2.2 Weighted Fair Queueing (WFQ)

El segundo algoritmo que se ha implementado es el *Weighted Fair Queuing*. El WFQ presenta un avance frente al RR introduciendo el concepto de prioridad. La prioridad ofrece la oportunidad de escoger una opción de split con mayor frecuencia frente a otras opciones, atendiendo a factores como, por ejemplo, la capacidad de los enlaces.

A modo de ejemplo, en un escenario con 2 *splits*, RR elegiría ambas opciones con la misma probabilidad, o, lo que es lo mismo, ambas opciones tendrían la misma prioridad. En WFQ podemos dar diferentes prioridades a cada opción. Por ejemplo, damos al primer *split* una prioridad=3 y al segundo *split* una prioridad=1. Esto significa que en media, de cada 4 veces (3+1), WFQ escoge 3 veces al *split* 1 y 1 vez al *split* 2. En términos de probabilidad, se escoge con una probabilidad de 0.75 al *split* 1 y de 0.25 al *split* 2. En la Figura 2.3 se muestra esta comparación.

Prio. Split 1	1
Prio. Split 2	1

Prio. Split 1	3
Prio. Split 2	1

RR		
Split	1	2
Pr	0,5	0,5

WFQ		
Split	1	2
Pr	0,75	0,25

Figura 2.3: Comparativa entre RR y WFQ

Aunque los resultados muestran una mejora frente RR, este algoritmo sigue sin ser el más eficiente. Esto se debe a que las prioridades se establecen a priori y no atendiendo al estado del sistema en cada *slot*. El WFQ sigue sin ser capaz de atender a los picos de tráfico de usuario en cada *slot* temporal, provocando un comportamiento no óptimo.

2.2.3 Backpressure (BP)

El tercer y último algoritmo que se ha implementado es el *Backpressure*. El algoritmo *Backpressure* fue introducido por Tassiulas y Ephremides [10] y ha sido objeto de estudio en los últimos años [11] por su habilidad para equilibrar las cargas de tráfico, mejorar el *throughput* y su simple implementación en redes de comunicaciones. El algoritmo está basado matemáticamente en teoría de Lyapunov [12]. Este algoritmo asegura la estabilidad de los sistemas de colas, adecuando las decisiones al estado de la mismas de manera "oportunista" en cada ejecución.

Como se ha mencionado se asume que el tiempo está ranurado y que se tiene un sistema de N colas conectadas por enlaces con cierta capacidad. El algoritmo *Backpressure* debe escoger en cada *slot* t una acción α que maximice el siguiente sumatorio:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} b_{i,j}(\alpha(t), S(t))W_{i,j}(t)$$

Para un *slot* t y una acción α concretos, se debe realizar el sumatorio del producto $b_{i,j}(\alpha(t), S(t))W_{i,j}(t)$ para todos los nodos del sistema. El término $b_{i,j}$ consiste en el tráfico de salida del nodo i seleccionado al nodo j seleccionado en el *slot* t concreto. Este término depende de dos variables: $\alpha(t)$ representa la opción de *split* seleccionada y $S(t)$ es una variable aleatoria que puede representar, por ejemplo, fluctuaciones de las capacidades previstas para los enlaces.

El termino $W_{i,j}$ representa el peso entre dos nodos. En primer lugar hay que tener en cuenta que en un mismo enlace pueden coexistir flujos de tráfico diferentes. Para cada flujo de tráfico c, se debe hallar el diferencial de colas entre ambos nodos que sigue la siguiente expresión:

$$W_{i,j}^{c}(t) = Q_{i}^{c}(t) - Q_{j}^{c}(t)$$

Para cada enlace, se escoge el peso $W_{i,j}$ correspondiente únicamente a un flujo c. Este es el que se introduce en la ecuación inicial a maximizar. Concretamente:

$$W_{i,j}(t) = \max_{c} [\max W_{i,j}^{c}(t), 0]$$

Finalmente, se evalúan todas las opciones α posibles y se selecciona el tráfico definitivo en cada enlace del sistema atendiendo a la opción que maximice la ecuación inicial. De este modo, la cantidad de tráfico que se traslada desde la cola i a la cola j para un flujo c en un determinado *slot* viene dada por la variable $\mu_{i,j}^{c}$ que se define a continuación:

$$\mu_{i,j}^{c} = \begin{cases} b_{i,j}(\alpha(t), S(t)), & \text{if } c \text{ maximizes the differental backlog} \\ 0, & \text{otherwise} \end{cases}$$

Con todo ello, una solución basada en *Backpressure* da preferencia a enviar tráfico desde las colas que se encuentran más llenas.

Este algoritmo se ha usado en diferentes escenarios en el pasado, poniendo de relieve que asegura la estabilidad de las colas a la vez que permite alcanzar valores altos de *throughput*.

Chapter 3

Implementación

La implementación consiste en el desarrollo de un entorno de simulación en Matlab en el que se modelan, mediante sistemas de colas, los nodos distribuidos y centralizados que componen la arquitectura vRAN, así como los enlaces que los conectan.

Como se ha mencionado anteriormente, para el funcionamiento del sistema, se asume un tiempo ranurado en *slots*. Al comienzo de cada *slot* se calcula el tráfico de entrada en el sistema. Después, se aplica el *scheduler* seleccionado para obtener los *splits* óptimos. De acuerdo a los *splits* seleccionados, se actualizan las colas del sistema. Finalmente en cada slot se almacena toda la información relativa al estado de las colas, retardo sufrido por los paquetes y selección de *split*.

Los parámetros básicos de configuración se fijan en el propio código y hacen referencia a parámetros como el tiempo de simulación en *slots*, el *scheduler* seleccionado y el número total de opciones de *split*.

3.1 Sistema

El sistema está basado en la arquitectura vRAN 5G que está formada por tres componentes principales: *Central Unit*, *Distributed Unit* y *Remote Unit*. A su vez, cada componente está formado por colas que procesan los paquetes y por enlaces que conectan cada elemento del sistema.

Para ilustrar mejor esta idea, en la Figura 3.1 se muestra un modelo simplificado de la arquitectura vRAN 5G. Este modelo básico es el punto de partida de la implementación y servirá de base para la evolución a escenarios más realistas. Aunque en este trabajo únicamente se abordará el enlace descendente, el modelado de la comunicación desde la RU a la CU sería similar.

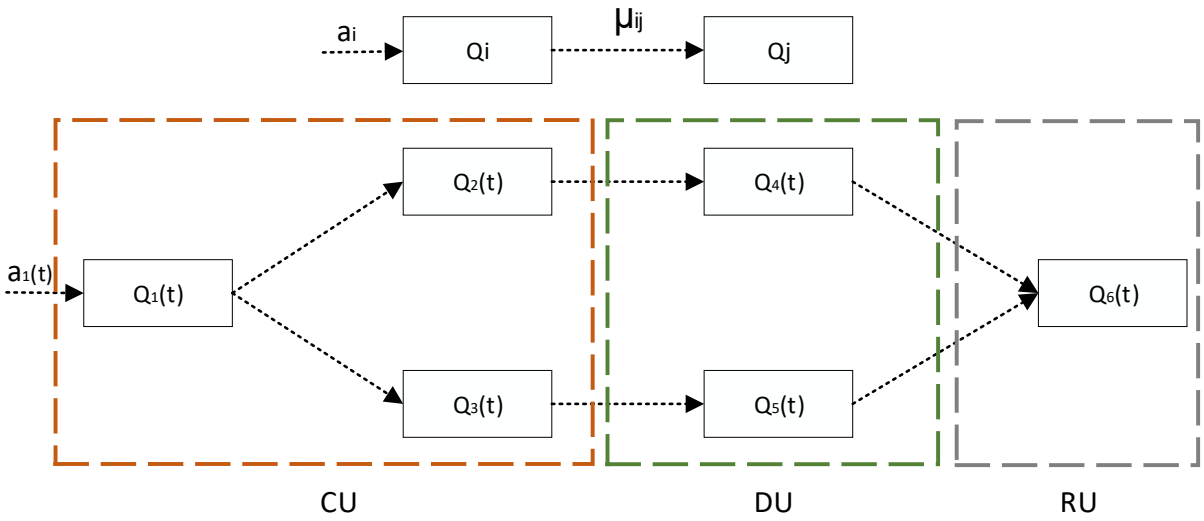


Figura 3.1: Modelo básico del sistema

En la Figura 3.1 se asume que únicamente se tienen 2 niveles de *split*. Como se puede ver, la CU está compuesta por una cola de entrada Q1 y una cola para cada nivel de *split*, Q2 y Q3 respectivamente. Por otro lado, la DU está formada por una cola que representa cada uno de los *split* y que actúa a la vez como cola de entrada y salida de este elemento. La diferencia del modelado de la CU y DU se debe a que una vez que un paquete se ha empezado a transmitir con un determinado nivel de *split* este no varía. De este modo, la decisión de *split* en la CU deberá mantenerse en la DU y de ahí que las colas de la DU sean a la vez de entrada y salida. Finalmente la RU se modela como una única cola que actuará de sumidero de tráfico.

3.1.1 Enlaces

Los paquetes que se transmiten entre las diferentes colas del sistema lo hacen a través de enlaces. Dependiendo de qué componentes comuniquen, los enlaces pueden pertenecer a una de las siguientes secciones:

- *Red Backhaul*: La conforman aquellos enlaces que conectan la cola inicial de la CU (a la cual llega el tráfico de entrada del sistema) con las colas que procesan los paquetes en la CU.
- *Red Midhaul*: La conforman aquellos enlaces que conectan las colas de la CU con la DU.

- *Red Fronthaul*: La conforman aquellos enlaces que conectan las colas de la DU con la RU.

Aunque en cada red puede haber múltiples enlaces, las colas tienen una única interfaz de entrada y una única interfaz de salida. Así, los enlaces representan de manera virtual las diferentes capacidades que puede tener el canal dependiendo de la opción de *split* seleccionada.

De esta manera, el *scheduler* utilizado deberá seleccionar qué enlaces de cada uno de los 3 segmentos se utilizan.

El número de combinaciones posible de selección de *splits* depende del número de opciones de split y del número de estaciones base. Sigue la siguiente expresión:

$$\#Combinaciones = (\#Split^3)^{\#EstacionesBase}$$

3.1.2 Capacidades

La asignación de las capacidades de los enlaces [paquetes/slot] que conforman el sistema siguen la idea propuesta por el 3GPP en [9]. Cada enlace representa una opción de *split* que, dependiendo de si es una opción más o menos centralizada tendrá asignada una capacidad diferente. En la Figura 3.2 se muestra un ejemplo para cuatro opciones de *split* con datos sintéticos.

	Ini - CU	CU - DU	DU - RU
Split 1	2	10	8
Split 2	4	10	6
Split 3	6	10	4
Split 4	8	10	2

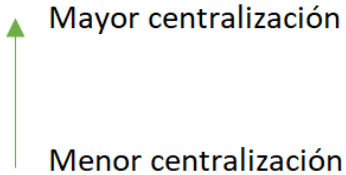


Figura 3.2: Capacidades [paquetes/slot] para un sistema con cuatro opciones de *split*

En la CU, la opción 1 es la más centralizada y en la que se procesan un mayor número de tareas. A mayor número de tareas por paquete, menor número de paquetes se procesan, siendo la capacidad final mucho menor. La capacidad de la CU va aumentando progresivamente hasta la opción 4 que es la más descentralizada y en la que se tratan un menor número de tareas, por lo que un mayor número de paquetes podrán ser procesados.

La asignación de capacidades en la DU se complementa con la de la CU. En la opción 1, la CU ya ha procesado la mayoría de las tareas, por lo que la DU apenas debe ocuparse de ninguna tarea y puede procesar un mayor número de paquetes, siendo la capacidad más alta. A medida que aumentamos el nivel de split, la DU se encarga de un mayor número de tareas y la capacidad final disminuye.

En el caso de las capacidades de la red *midhaul* que conecta la unidad centralizada con la unidad distribuida, se considera un canal dedicado de fibra óptica o microondas, de manera que todos los enlaces

tienen la misma capacidad. Además, este trabajo no está centrado en esta sección de la red, por lo que consideramos capacidades altas que no interfieran en la estabilidad y retardo del sistema.

3.1.3 Tráfico de entrada

La generación del tráfico de entrada se lleva a cabo en la función `trafico_entrada`. En este trabajo el tráfico de entrada sigue una distribución de Poisson aunque se ha preparado la función para soportar tráficos con diferentes distribuciones como uniforme o binomial. El tráfico se mide en paquetes por *slot* y su implementación se describe en la Figura 3.3.

```
1 function [d_poisson]=trafico_entrada()
2
3 d_uniforme=randi([10,20]); %Uniform distribution
4
5 d_poisson=poissrnd(lambda); %Poisson distribution
6 lambda=1500;
7
8 d_binomial=binornd(n, p); %Binomial distribution
```

Figura 3.3: Generación del tráfico de entrada

3.1.4 Actualización de las colas

De acuerdo a la elección de los *splits* por parte del *scheduler* utilizado, se deben actualizar las colas del sistema. Las colas siguen un esquema FIFO, de manera que los primeros paquetes que llegan son también los primeros paquetes que salen. La actualización sigue la expresión 3.1, donde el estado de la cola en el slot $t+1$ es la diferencia del estado de la cola en el slot t y el tráfico de salida $b_k(t)$ (siempre que la diferencia no sea un valor negativo) más el tráfico de entrada en el slot t $a_k(t)$:

$$Q_k(t+1) = \max\left[Q_k(t) - b_k(t), 0\right] + a_k(t) \tag{3.1}$$

A nivel de código, esta actualización se lleva a cabo en la función `actualizar_colas`. En primer lugar se obtiene el tráfico de entrada y de salida de acuerdo a la capacidad del enlace seleccionado, para posteriormente aplicar la expresión 3.1, tal como se muestra en la Figura 3.4.

```
function [] = actualizar_colas()

%%Initial
if capacidad_backhaul(split)< qinicial_k(slot)
    binicial_k(slot)=capacidad_backhaul(split);
else
    binicial_k(slot)=qinicial_k(slot);
end

qinicial(slot+1)=max([qinicial(slot)-binicial(slot), 0])+ainicial(slot);

%%CU
if i==split
    acu_k(slot+1,i)=binicial_k(slot);
else
    acu_k(slot+1,i)=0;
end

if capacidad_midhaul(split_enlace)< qcu_k(slot, split_enlace)
    bcu_k(slot, split_enlace)=capacidad_midhaul(split_enlace);
else
    bcu_k(slot, split_enlace)=qcu_k(slot,split_enlace);
end

for i=1:1:split_enlace_total
    if i≠split_enlace
        bcu_k(slot,i)=0;
    end
end

for i=1:1:cu_total
    qcu(slot+1,i)=max([qcu(slot,i)-bcu(slot,i), 0])+acu(slot,i);
end

%%DU
for i=1:1:split_enlace_total
    adu_k(slot+1,i)=bcu_k(slot,i);
end

if capacidad_fronthaul(split_du) < qdu_k(slot,split_du)
    bdu_k(slot,split_du)=capacidad_fronthaul(split_du);
else
    bdu_k(slot,split_du)=qdu_k(slot,split_du);
end

for i=1:1:split_du_total
    if i≠split_du
        bdu_k(slot,i)=0;
    end
end

for i=1:1:du_total
    qdu(slot+1,i)=max([qdu(slot,i)-bdu(slot,i), 0])+adu(slot,i);
end
```

Figura 3.4: Actualización de las colas

3.2 Schedulers

Antes de la ejecución de la simulación, se elige el *scheduler* a utilizar. A continuación se muestra la implementación de los diferentes *schedulers* que se han utilizado.

3.2.1 Round Robin

Se trata del *scheduler* más sencillo de implementar, puesto que simplemente debe escoger de manera aleatoria una opción de *split*.

La distribución uniforme en Matlab se obtiene fácilmente con la función `randi([imin, imax])` que devuelve un entero escalar pseudoaleatorio en el intervalo `imin, imax`. Este proceso se repite tantas veces como estaciones base haya en el escenario. La Figura 3.5 muestra la implementación de *scheduler* RR.

```
1  if scheduler_sel==1 %Round Robin
2      for estacion=1:1:estacion_total
3          split_optimo(estacion)= randi([1,split_total]);
4
5          split_enlace_optimo(estacion)= randi([1,split_enlace_total]);
6
7          split_du_optimo(estacion)= randi([1,split_du_total]);
8
9          eleccion_splits(slot, :, estacion)=[split_optimo(estacion) ...
               split_enlace_optimo(estacion) split_du_optimo(estacion)];
10     end
```

Figura 3.5: *Round Robin*

3.2.2 Weighted Fair Queuing

Este algoritmo presenta una mayor complejidad que el *Round Robin*, puesto que introduce el concepto de prioridad que permite poder elegir unos *splits* con mayor probabilidad que otros. En primer lugar se define un vector en el que se introducen las prioridades para cada *split*, tal y como se muestra en la Figura 3.6.

Después, se incluye en otro vector conjunto cada opción de *split* tantas veces como hayamos definido en la prioridad.

Haciendo uso de nuevo de la función `randi([imin, imax])` de Matlab, se elige de manera pseudoaleatoria uno de los elementos de este vector final. Aquella opción que aparezca más veces tiene más posibilidades de ser seleccionada.

```
1  %Priorities
2
3  prioridades_cu=[prioridad_split1_cu prioridad_split2_cu];
4  prioridades_enlace=[prioridad_split1_enlace prioridad_split1_enlace];
5  prioridades_du=[prioridad_split1_du prioridad_split2_du];
```

Figura 3.6: WFQ: Definición de prioridades

```
if scheduler_sel==2 %WFQ
    for estacion=1:1:estacion_total
        k=1;
        for i=1:1:split_total
            tope=prioridades_cu(i);
            for j=1:1:tope
                vector_cu(k)=i;
                k=k+1;
            end
        end
        split_optimo(estacion)=vector_cu(randi(length(vector_cu),1,1));

        k=1;
        for i=1:1:split_du_total
            tope=prioridades_du(i);
            for j=1:1:tope
                vector_du(k)=i;
                k=k+1;
            end
        end
        split_du_optimo(estacion)=vector_du(randi(length(vector_du),1,1));

        k=1;
        for i=1:1:split_enlace_total
            tope=prioridades_enlace(i);
            for j=1:1:tope
                vector_en(k)=i;
                k=k+1;
            end
        end
        split_en_optimo(estacion)=vector_en(randi(length(vector_en),1,1));
    end
```

Figura 3.7: WFQ: Selección de *splits* óptimos

3.2.3 Backpressure

La implementación del algoritmo *Backpressure* es la más compleja, puesto que tiene que evaluar en tiempo real todas las opciones posibles para elegir los *splits* óptimos. Para cada opción, se llama a la función `backpressure` que se muestra en la Figura 3.8.

Este resultado se guarda en una variable llamada `sumatoriobackpressure` que contiene tres dimensiones correspondientes a los tres niveles de *split* y una cuarta dimensión para la estación base en la que se encuentra la simulación.

Una vez evaluadas todas las opciones, se busca el valor máximo en la variable `sumatoriobackpressure` y se obtienen los índices correspondientes a dicho valor, que corresponden a la combinación de *splits* en la red *backhaul, midhaul y fronthaul*. Esto se consigue vectorizando la variable `sumatoriobackpressure` y obteniendo los índices correspondientes al valor máximo mediante la función `ind2sub`.

```
function [sumatorio] =backpressure()

sumatorio=0;
for i=1:1:inicial_total
    for j=1:1:cu_total
        peso_colas=max([qinicial_k(slot, i)-qcu_k(slot,j), 0]);
        sumatorio=sumatorio+binicial_k(slot,i)*peso_colas;
    end
end

for i=1:1:cu_total
    peso_colas=max([qcu_k(slot, i)-qdu_k(slot,i), 0]);
    sumatorio=sumatorio+bcu_k(slot, i)*peso_colas;
end

for i=1:1:du_total
    for j=1:1:ru_total
        peso_colas=max([qdu_k(slot, i)-qru_k(slot,j), 0]);
        sumatorio=sumatorio+bdu_k(slot, i)*peso_colas;
    end
end

end
```

Figura 3.8: Función backpressure

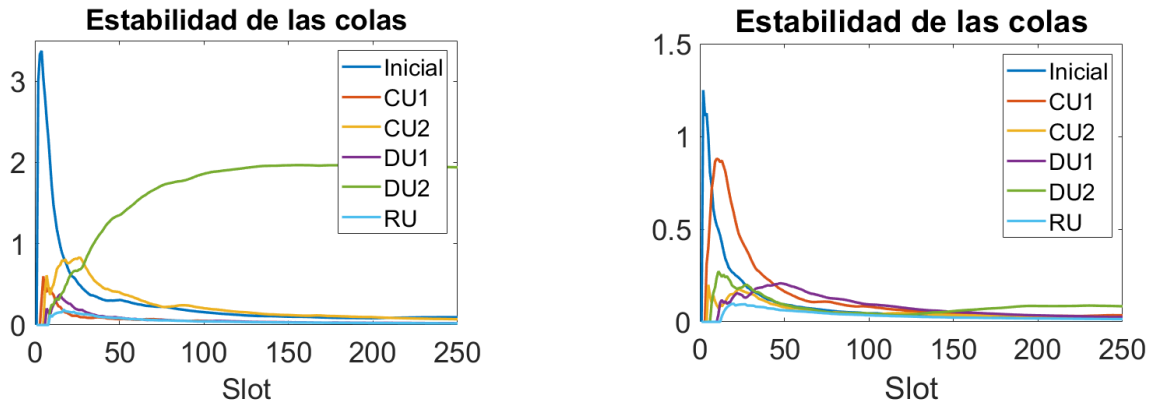
```
if scheduler_sel==3 %Backpressure
    %Get the optimal splits - Backpressure
        sumatorio_backpressure_aux2=sumatorio_backpressure(:,:,:,estacion);
        num_max=max(sumatorio_backpressure_aux2(:));

    %Find the maximum value
        [row,col]=find(sumatorio_backpressure_aux2(:)==num_max);
        elec=datasample(row,1);

    %Get the index of the maximum value
        [split_optimo, split_du_optimo, ...
            split_enlace_optimo]=ind2sub(size(sumatorio_backpressure_aux2),elec);
        eleccion_splits(slot, :, estacion)= [split_optimo split_enlace_optimo ...
            split_du_optimo];
```

Figura 3.9: Backpressure: Obtener los *splits* óptimos

3.3 Métricas

Para comparar el funcionamiento de los *schedulers* en diferentes escenarios se han utilizado tres tipos de métricas: el retardo, la estabilidad de las colas y la selección de *splits*. A continuación se define cada una de ellas y se muestra un ejemplo de representación e interpretación.

3.3.1 Estabilidad de las colas



(a) Colas estables

(b) Colas estables y una cola inestable

Figura 3.10: Representación de la estabilidad de las colas

La primera métrica empleada es la estabilidad de las colas, que se puede medir de diferentes maneras. La que se emplea en este trabajo es la *mean rate stability*. Diremos que una cola presenta *mean rate stability* si cumple la Ec. 3.2 Es decir, si el valor medio temporal (*time average*) de la esperanza matemática de su ocupación tiende a 0. Intuitivamente, esta definición nos indica que una cola es estable si no crece indefinidamente.

$$\lim_{t\to\infty} \frac{E\{|Q(t)|\}}{t} = 0 \tag{3.2}$$

A modo de ejemplo en la Figura 3.10a se muestra la estabilidad de varias colas en una simulación de 250 *slots* a las que se ha aplicado el *scheduler Round Robin*. Como se puede ver, todas las colas tienden a 0, por lo que todas serían estables. Por otra parte en la Figura 3.10b se muestra la estabilidad de varias colas en una simulación con el mismo número de *slots* y mismo *scheduler*, con la diferencia de que la segunda cola asociada a la DU no tiende a 0 sino a 2, por lo que se considera inestable.

3.3.2 Probabilidad de selección de *split*

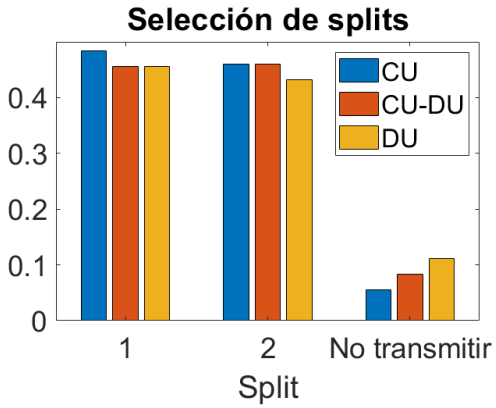


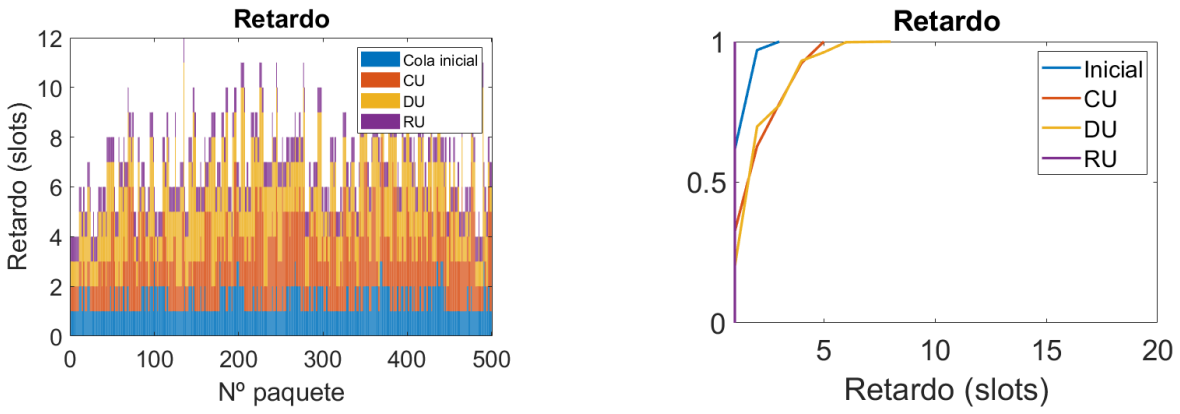
Figura 3.11: Selección de *splits*

La segunda de las métricas consiste en la probabilidad con la que se selecciona uno de los diferentes niveles de cada split en cada segmento de la red. Tendremos información referente a la selección de split en la red *backhaul* (enlace cola inicial - CU), la red *midhaul* (enlace CU-DU) y la red *fronthaul* (enlace DU-RU).

En la Figura 3.11 se muestra la selección de *split* para cada segmento de la red. Como es lógico, la suma de todas las probabilidades con las que se selecciona cada split es 1. Además de las opciones de *split* de las que ya se ha hablado en el trabajo, se incluye una opción adicional denominada 'No transmitir'. Como indica su nombre, no se procesan paquetes en el segmento de la red en la que se seleccione. Esta opción solo está implementada en el algoritmo *Backpressure*.

La información se guarda en una variable de llamada `eleccion_splits`. Una vez finalizada la simulación la función `representacion_splits()` se encarga de contar las veces que se ha seleccionado cada uno y dividirlo entre el número total de slots para obtener las probabilidades.

3.3.3 Retardo



(a) Gráfico de barras

(b) Función de distribución acumulativa empírica

Figura 3.12: Retardo de los paquetes en el sistema

La tercera métrica es el retardo, el cual describe el número de *slots* que permanece un paquete en el sistema. Para ello se obtiene el tiempo que permanece un paquete en cada cola del sistema y se suman los tiempos de cada cola por el que ha pasado el paquete para obtener el retardo total. La implementación en Matlab es más compleja en comparación con el resto de métricas.

En primer lugar, se asigna un identificador a cada paquete que entra en la cola inicial. Después se obtiene el *slot* de entrada y el *slot* de salida de cada paquete y se almacenan en una variable. El resultado de la resta entre el *slot* de salida y el *slot* de entrada nos da el retardo como tal. Una vez tenemos las listas de retardo para todas las colas del sistema, obtendremos la suma de todas ellas para cada paquete, obteniendo una lista final que es la que posteriormente se plasma en las gráficas de resultados.

En la Figura 3.12a, se muestra el retardo de 500 paquetes que han pasado por el sistema. Todos los paquetes deben pasar por la cola inicial (azul), una cola de la CU (rojo), una cola de la DU (naranja) y una cola de la RU (morado) para abandonar el sistema.

Para hacer más fácil la comparativa del retardo en diferentes escenarios, se utilizará la Función de Distribución Acumulativa Empírica (ECDF) mostrada en la Figura 3.12b que se obtiene fácilmente con la función `[f,x]=ecdf(y)` de Matlab.

Chapter 4

Resultados

En esta sección se muestran los resultados obtenidos de las diferentes simulaciones que se han realizado sobre la arquitectura 5G vRAN bajo diferentes condiciones.

Se han definido tres escenarios, en los que se pone a prueba el rendimiento del algoritmo *Backpressure* para la selección de *splits* comparando su comportamiento con otros algoritmos.

El primer escenario, llamado **escenario 0**, tiene como objetivo comprobar si la implementación del sistema es correcta y demostrar la ventaja de *Backpressure* frente a otros *schedulers*. El segundo escenario o **escenario 1** tiene una función más analítica, pues se pone a prueba por primera vez al algoritmo *Backpressure* en un escenario con capacidades reales, donde además se realiza un barrido de la capacidad de la DU y se evalúa al algoritmo frente a diferentes patrones de tráfico. Por último, el **escenario 2** incrementa la complejidad del sistema añadiendo más estaciones base y limitando la capacidad de cómputo de la *Central Unit*, de manera que no todas las opciones de selección de *split* sean posibles.

Para analizar los diferentes escenarios se han usado las métricas descritas en el capítulo anterior: la estabilidad de las colas, la selección de *splits* y el retardo.

A continuación se muestran en detalle los tres escenarios dispuestos, sus casos concretos, objetivos y resultados.

4.1 Escenario 0

En el primer escenario, el principal objetivo es comprobar que la implementación de la arquitectura V-RAN en Matlab funciona correctamente. Para ello se utilizan datos sintéticos y un escenario simple que sirve de base para las implementaciones más realistas que se dan en escenarios posteriores. Además, se pretende comparar el funcionamiento de los tres *schedulers* que se encargan de la selección de *splits*: *Round Robin*, *Weighted Fair Queuing* y *Backpressure*.

En la Figura 4.1 se puede ver un esquemático del sistema que consta de una estación base, dos opciones de *split* y capacidades de los enlaces con valores sintéticos.

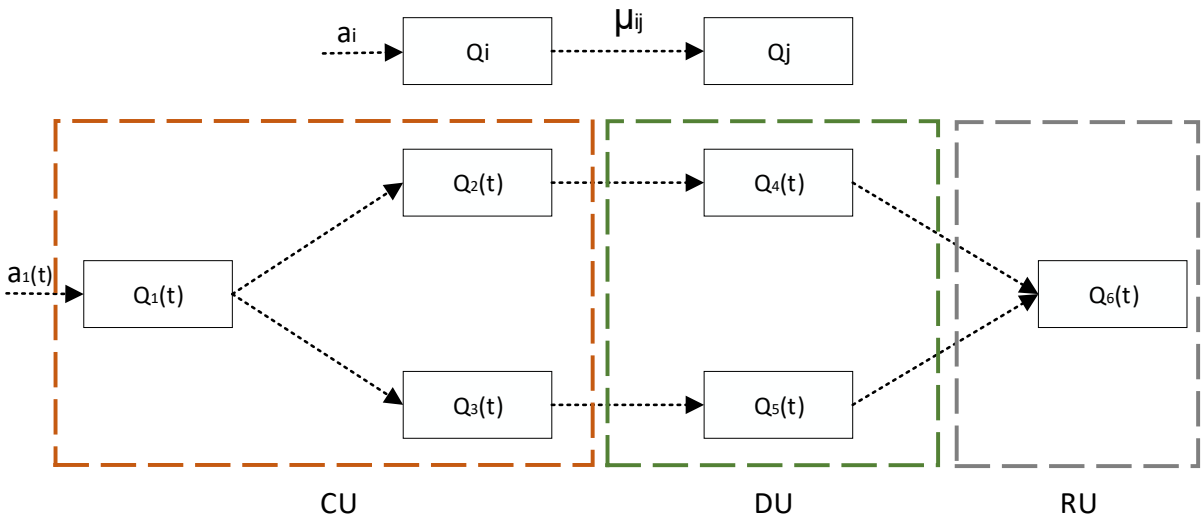


Figura 4.1: Esquema escenario 0

Tanto en este escenario como en los siguientes, las capacidades de los enlaces de la red *midhaul* y la RU tienen valores muy altos en comparación al resto de enlaces para que no interfieran en los valores de retardo y estabilidad del sistema.

Dentro de este escenario, se varían las capacidades de los enlaces y las tasas de entrada dando lugar a los casos concretos que se muestran a continuación.

4.1.1 Caso 1: Capacidades homogéneas

El primer caso consiste en asignar la misma capacidad a todos los enlaces del sistema, tal y como se muestra en la Tabla 4.1, para después aplicar cada uno de los *schedulers* implementados para la selección de *split*. *Round Robin* elige los *splits* con la misma probabilidad, *Weighted Fair Queuing* lo hace de manera proporcional a las capacidades de los enlaces, pero al tener todos los enlaces la misma capacidad se comporta igual que *Round Robin*; y *Backpressure* lo hace en tiempo real, lo que debería darle una ligera ventaja frente a los algoritmos anteriores. En este escenario se opta por una tasa de entrada baja que sigue una distribución de Poisson con tasa λ de valor 4 paquetes por *slot*.

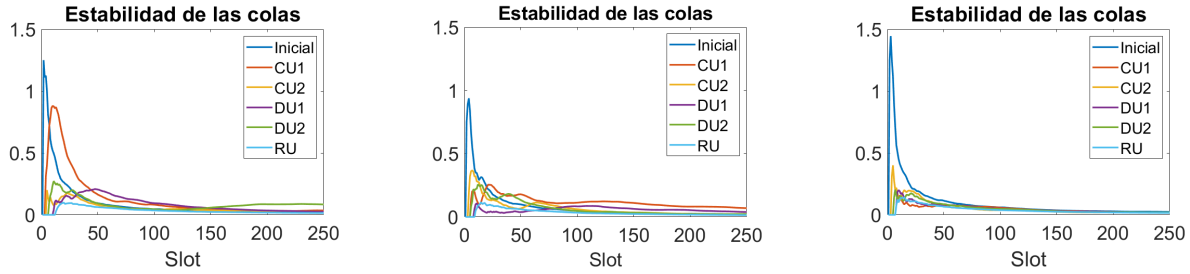
Tabla 4.1: Capacidades de los enlaces y tasa de entrada para el Caso 1

Capacidad enlaces (`pkts/s`)	CU	CU-DU	DU	RU
Split 1	5	5	5	5
Split 2	5	5	5	5
Tasa de tráfico (`pkts/s`)	Poisson $\lambda = 4$			

La Figura 4.2 representa la estabilidad del sistema para cada *scheduler*, usando el *time average* de las colas a lo largo del tiempo. Se puede apreciar un comportamiento estable para todos ellos, pues el valor de la función tiende a 0. La tasa de entrada del sistema es de 4 paquetes por *slot*, menor que todas las capacidades de los enlaces, por lo que no es un valor lo suficientemente grande para saturar las unidades de procesado.

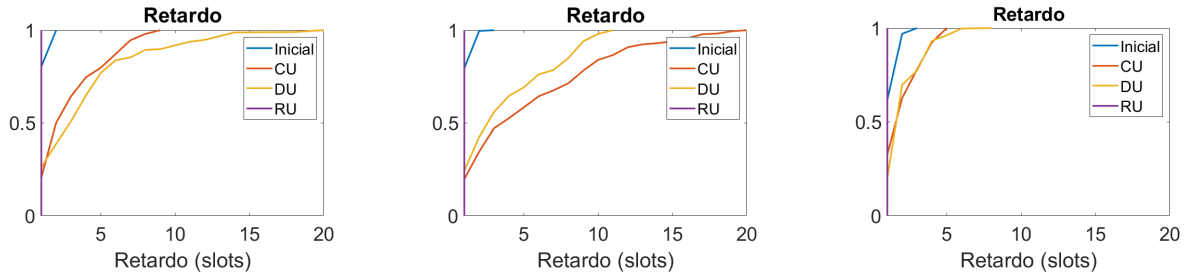
Después, en la Figura 4.3 se muestra la Función de Distribución Acumulativa Empírica del retardo de los paquetes. Se puede apreciar cómo *Round Robin* y *Weighted Fair Queuing* tienen unos valores de retardo muy similares ya que, debido a que todos los enlaces tienen la misma capacidad, ambos se comportan como el mismo algoritmo. Mientras las Figuras 4.3a y 4.3b muestran retardos de hasta 20 *slots* en algunos segmentos del sistema, en la Figura 4.3c ningun tipo de retardo supera los 8 *slots*, lo que demuestra la ventaja de *Backpressure* incluso en escenarios con capacidades homogéneas y baja tasa de tráfico.

Respecto a la selección de las opciones de *split*, representado en la Figura 4.4, el comportamiento es el esperado para *RR* y *WFQ*, puesto que se escogen todas las opciones con una probabilidad similar, tal y como están preconfigurados. En el caso de *Backpressure*, se puede concluir que a capacidades homogéneas y una tasa de tráfico baja, la selección de *split* se reparte de manera similar en todos los segmentos de la red. Además, es interesante el hecho de que la opción de no transmitir paquetes apenas se selecciona un 10% de las veces, como se muestra en la Figura 4.4c. A la vista de los resultados queda patente que el comportamiento oportunista del algoritmo de *Backpressure* le da ventaja frente a los pre-configurados. En concreto, se ve que estadísticamente la selección de split con *Backpressure* es similar a los otros algoritmos, pero se consigue un retardo en torno a la mitad.



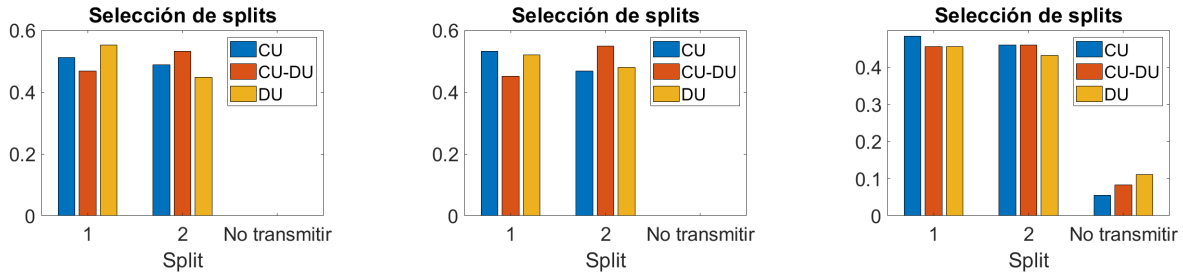
(a) Round Robin (b) Weighted Fair Queuing (c) Backpressure

Figura 4.2: Estabilidad de las colas



(a) Round Robin (b) Weighted Fair Queueing (c) Backpressure

Figura 4.3: Retardo de los paquetes



(a) Round Robin (b) Weighted Fair Queueing (c) Backpressure

Figura 4.4: Selección de *splits*

4.1.2 Caso 2: Capacidades heterogéneas con poco tráfico

En el segundo caso, se analiza el comportamiento de los tres *schedulers* tomando unas capacidades más dispares, que se muestran en la Tabla 4.2. El objetivo es comenzar a destacar diferencias más notables en los resultados de cada algoritmo. En este caso, al ser las capacidades de los enlaces diferentes, el WFQ escogerá los *splits* en cada *slot* de manera proporcional a las capacidades. Esto significa que seleccionará con mayor probabilidad la opción 2 que la opción 1, tratando de imitar al algoritmo *Backpressure*.

La tasa de entrada se mantiene con un con un valor medio λ de 4 paquetes por *slot*, favoreciendo la estabilidad del sistema.

Tabla 4.2: Capacidades de los enlaces y tasa de entrada para el Caso 2

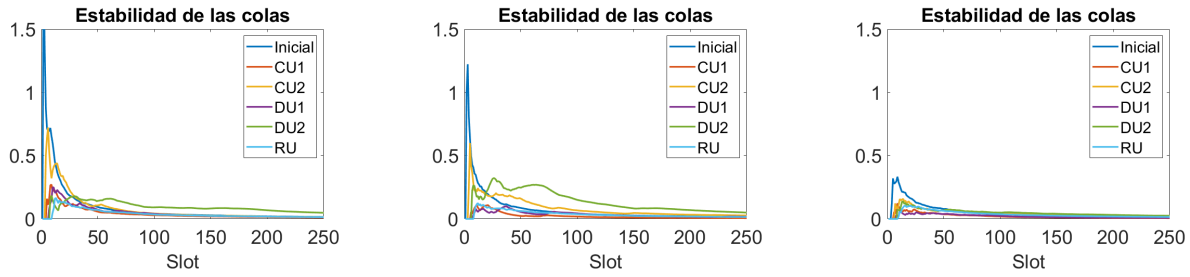
Capacidad enlaces (`pkts/s`)	CU	CU-DU	DU	RU
Split 1	5	20	15	20
Split 2	15	20	5	20
Tasa de tráfico (`pkts/s`)	Poisson $\lambda = 4$			

La Figura 4.5 muestra la estabilidad de las colas para cada *scheduler*. En todos los casos las funciones de estabilidad tienden a 0, por lo que todas las colas son estables. Sin embargo, se empieza a apreciar una pequeña mejoría de *Backpressure* con respecto al resto, pues prácticamente desde el comienzo de la simulación la función no muestra picos en la función de estabilidad, algo que sí ocurre para *Round Robin* y *Weighted Fair Queuing*.

La Figura 4.6 muestra el retardo de los paquetes que pasan por el sistema. *Backpressure* sigue teniendo unos resultados de retardo mucho mejores que *Round Robin* y *Weighted Fair Queuing*, el cual, a pesar de escoger los *splits* de manera proporcional a las capacidades de los enlaces, no consigue diferenciarse aún de *Round Robin*.

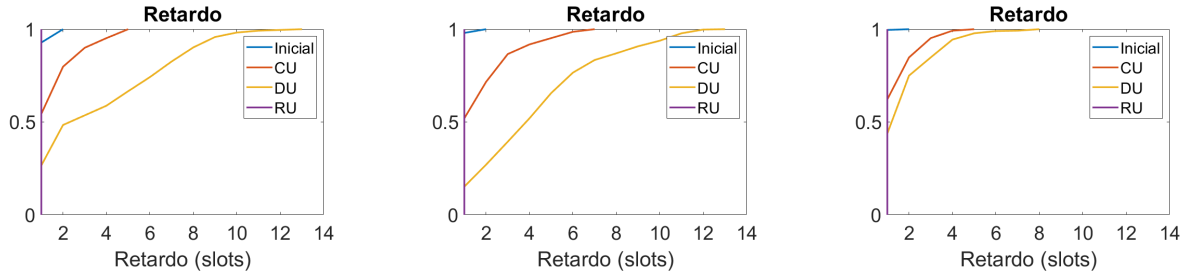
Por último en la Figura 4.7 se muestra la selección de los *splits*. *Round Robin* y *Weighted Fair Queuing* tienen los resultados esperados, pues la probabilidad de selección está preconfigurada. Por su parte, *Backpressure* escoge con una mayor probabilidad (alrededor del 65 %) la opción 2 que la opción 1 (alrededor de un 35 %). Este resultado muestra cómo, en un escenario con capacidades heterogéneas y con un tráfico que no comprometa la estabilidad de las colas, el algoritmo tiende a escoger con mayor probabilidad los enlaces con mayor capacidad en la CU mientras que escoge también la cola asociada a

este enlace en la DU, evitando que un mayor número de paquetes queden esperando en el sistema y, por tanto, reduciendo el retardo.



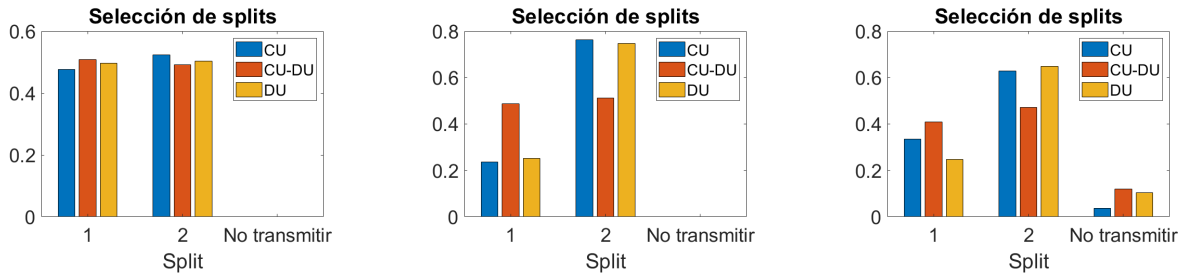
(a) Round Robin (b) Weighted Fair Queuing (c) Backpressure

Figura 4.5: Estabilidad de las colas



(a) Round Robin (b) Weighted Fair Queueing (c) Backpressure

Figura 4.6: Retardo de los paquetes

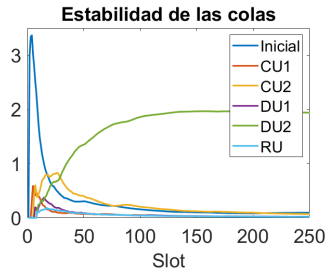


(a) Round Robin (b) Weighted Fair Queueing (c) Backpressure

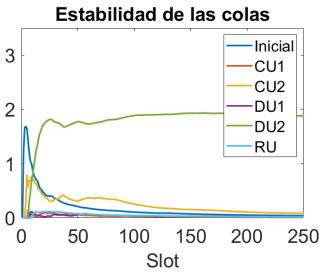
Figura 4.7: Selección de *splits*

4.1.3 Caso 3: Capacidades heterogéneas con tráfico alto

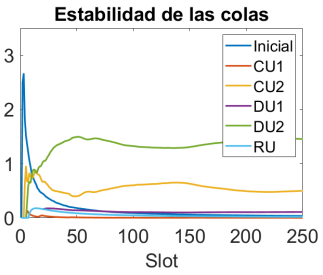
En el tercer caso se mantienen las capacidades heterogéneas de los enlaces definidas en la sección anterior, pero se añade un nivel más de exigencia al sistema aumentando la tasa de entrada a 9 paquetes por *slot*. Se pretende observar el comportamiento de los *schedulers* en un escenario que favorece la inestabilidad de las colas.



(a) Round Robin

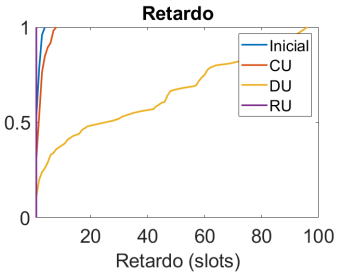


(b) Weighted Fair Queuing

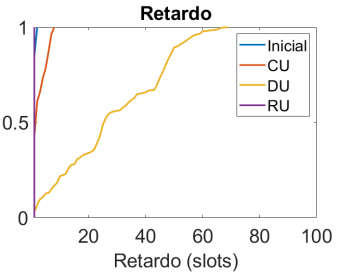


(c) Backpressure

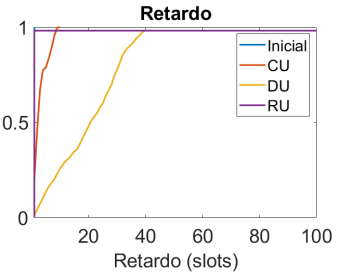
Figura 4.8: Estabilidad de las colas



(a) Round Robin

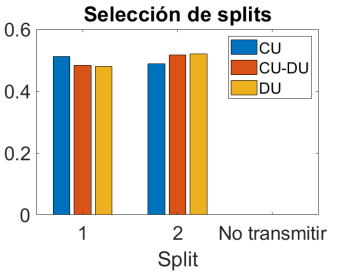


(b) Weighted Fair Queueing

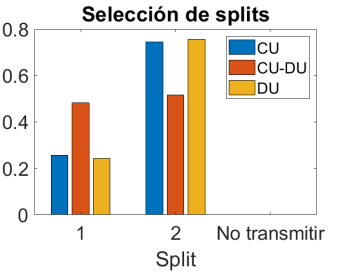


(c) Backpressure

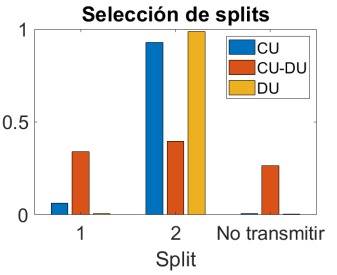
Figura 4.9: Retardo de los paquetes



(a) Round Robin



(b) Weighted Fair Queueing



(c) Backpressure

Figura 4.10: Selección de *splits*

Por primera vez en el **escenario 0**, aparecen colas inestables. Debido a la alta tasa del tráfico de entrada, en la Figura 4.8 se puede apreciar que en los tres *schedulers* la cola asociada a la DU de la opción 2 es inestable. Esto se debe a que la cola asociada a la CU de la opción 2 tiene una capacidad mucho más elevada que la de la DU, provocando que a esta última lleguen muchos paquetes y, debido a su baja capacidad, no sea capaz de procesarlos todos.

El retardo aumenta considerablemente con respecto a los anteriores casos, llegando a cantidades de más de 50 *slots* tal y como se aprecia en la Figura 4.9. Este retardo es mucho mayor en la DU, por el mismo motivo por el que las colas son inestables. Por primera vez se aprecia una clara ventaja del algoritmo WFQ frente a RR, demostrando que, a mayor tasa de tráfico, más eficaz es el segundo algoritmo con respecto al primero. *Backpressure* sigue demostrando controlar mejor el retardo de los paquetes que pasan el sistema, pero en la Figura 4.9c se puede apreciar una pequeña desventaja de este algoritmo: la curva referente a la DU y a la RU no llegan completamente a 1. Esto se debe a que *Backpressure* considera

que una cola con una cantidad cercana pero mayor que 0 paquetes es estable, por lo que se centrará en procesar paquetes del resto de colas, quedando los primeros 'atascados' en el sistema. Aunque son muy pocos paquetes los afectados con respecto al total, en algunos casos podría ser un problema.

En cuanto a la selección de *splits*, la Figura 4.10 indica que *Backpressure* escoge con una probabilidad cercana a 1 la opción 2 de *split* para la sección de la CU y la DU, lo que significa que, a mayor tasa de tráfico, el algoritmo opta por una mayor centralización de procesado de recursos. La selección para el *split* del enlace está muy repartida entre las tres opciones. El motivo es que si se seleccionase siempre la opción 2, el retardo de los paquetes sería aún mayor, pues la DU ya está saturada al tener una capacidad de procesado muy baja. Por ello, *Backpressure* alterna entre la opción de 'no transmitir' y la opción 1 que, en realidad, también es una opción de no enviar ya que las colas de la opción 1 están vacías. Es importante recalcar que, como en este caso, ante dos opciones que procesan el mismo número de paquetes, el algoritmo escoge una de ellas de manera aleatoria.

4.2 Escenario 1

El objetivo del **escenario 1** es poner a prueba el rendimiento del algoritmo *Backpressure* en un sistema más realista. En este escenario ya no se incluyen los otros algoritmos, ya que se prevé que su comportamiento será muy pobre.

En la Figura 4.11 se muestra un esquema del sistema implementado, que consta de una estación base, los ocho niveles de *split* previstos por el 3GPP [9] y capacidades con valores reales.

Para la asignación de estas capacidades de cómputo, se ha recurrido a [13], donde se indican los valores de complejidad computacional medidos en Gigaoperaciones por segundo (GOPS) que se necesitan para procesar cada una de las capas que conforman las comunicaciones móviles. De esta manera, se puede obtener la complejidad computacional para cada uno de los ocho niveles de *split* tanto en la CU como en la DU. En la primera columna de la Tabla 4.3 se muestra la complejidad computacional que conlleva procesar las funciones asociadas a cada nivel de *split* en la CU. Por ejemplo, C-RAN es el nivel más centralizado y, por tanto, del que más funciones se encarga y mayor capacidad de procesado necesita.

Por otra parte, se han tomado datos del enlace radio de estaciones base de diferentes tamaños, desde 6 hasta 100 PRBs, tales como el número de bits por *slot*, Mbps y paquetes por *slot*. De esta lista se toman los valores máximos, para cada tamaño de estación base, de la cantidad del número de paquetes o bits por *slot* que se procesan para la opción 1 (C-RAN). Este valor se multiplicará por el valor de la segunda columna de la Tabla 4.3 (columnas de la izquierda) que muestra la relación entre la capacidad de procesado que se requiere para procesar las funciones y la capacidad de procesado real de la CU medida también en GOPS. En concreto se ha usado una CU capaz de procesar 20 GOPS, por lo que, en el caso de C-RAN la relación es de 20.9/20= 0.96. Si la CU fuera capaz de procesar 25 GOPS, la relación sería 20.9/25=0.836. Una vez obtenidos los datos para la primera opción de *split*, se aplica de manera proporcional al resto de opciones.

Por último, para hallar las capacidades en la DU se toma la complejidad computacional que requiere procesar todas las funciones del enlace radio (21 GOPS) y se halla la diferencia con la complejidad computacional de las opciones de la CU, dando los valores de la Tabla 4.3 (columnas de la derecha). Al igual que en la CU, se tendrá en cuenta la capacidad nominal de la DU en GOPS para obtener las capacidades de los enlaces en paquetes por *slot*.

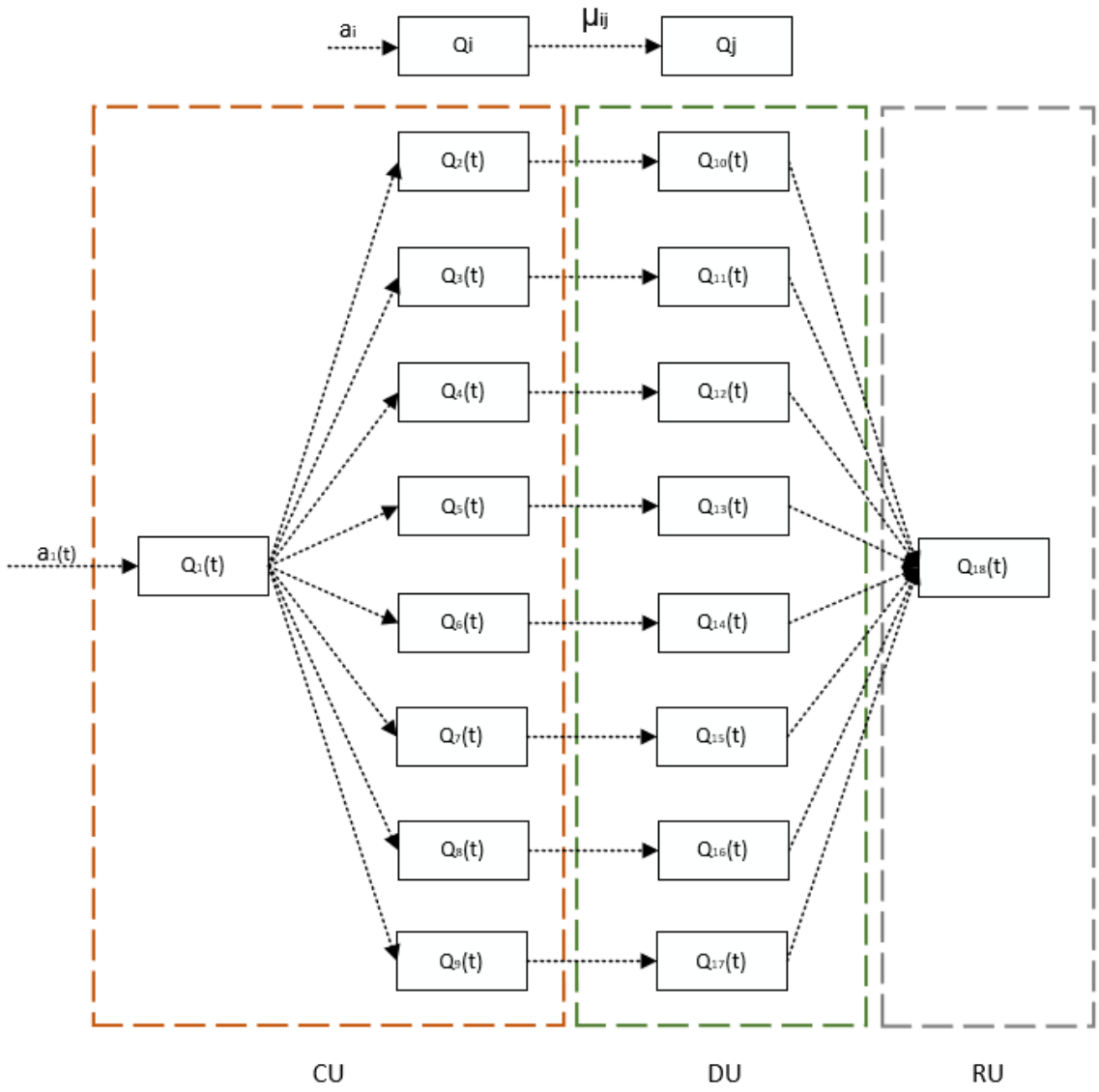


Figura 4.11: Esquemático del escenario 1

Tabla 4.3: Complejidad computacional en GOPS de la CU y DU para una estación base con ancho de banda de 20 MHz y capacidad de 20 GOPS

	Ratio relativo a capacidad			
	CU		DU	
C-RAN	20.9	0.96	0.1	180
Intra-PHY	16	1.25	5	3.6
MAC/PHY	10.7	1.87	10.3	1.75
Intra-MAC	8.7	2.3	12.3	1.46
RLC/MAC	6.7	2.99	14.3	1.26
Intra-RLC	5.7	3.51	15.3	1.18
PDCP/RLC	4.7	4.26	16.3	1.1
RRC/PDCP	2.7	7.41	18.3	0.98

Para este escenario se fija una CU de 20 GOPS y se hace un barrido en la capacidad de la DU, tomando

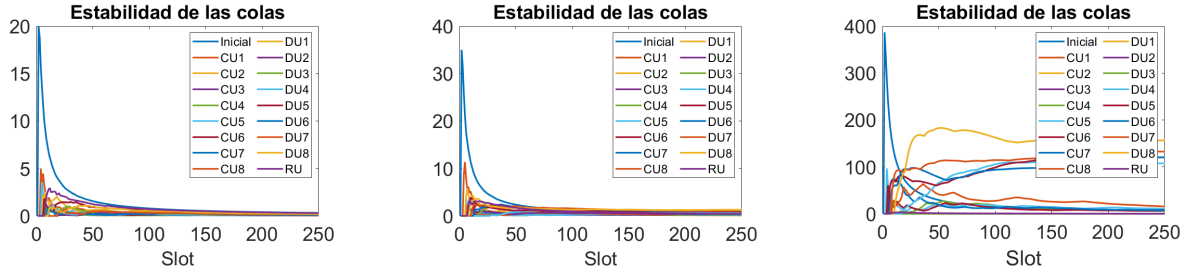
valores de entre un 10% y un 90% de la capacidad de la CU, dando lugar a tres casos:

- Caso 1: Capacidad DU = 0.10 · Capacidad CU = 2 GOPS
- Caso 2: Capacidad DU = 0.50 · Capacidad CU = 10 GOPS
- Caso 3: Capacidad DU = 0.90 · Capacidad CU = 18 GOPS

Las capacidades de los enlaces para cada caso se muestran en el Anexo A. Además para cada uno de los tres casos se evaluarán tres tasas de entrada diferentes:

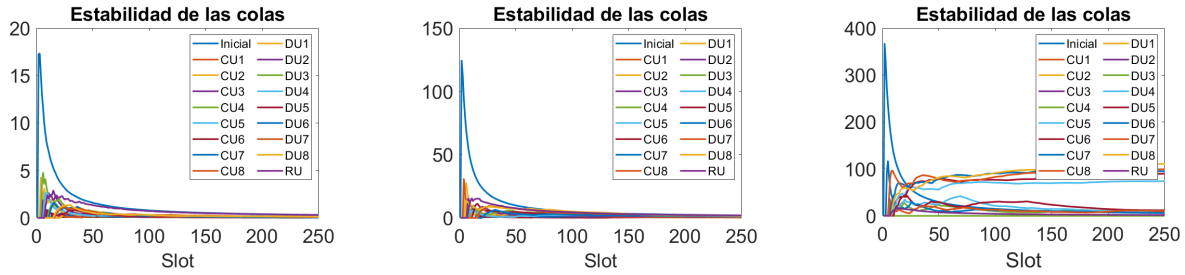
- Tráfico A: Distribución de Poisson con tasa λ = 80 paquetes por *slot*
- Tráfico B: Distribución de Poisson con tasa λ = 500 paquetes por *slot*
- Tráfico C: Distribución de Poisson con tasa λ = 1500 paquetes por *slot*

En las Figuras 4.12 4.13 y 4.14 se muestran los resultados de estabilidad para cada caso concreto. Se puede observar que, para un tráfico de entrada bajo como es el tráfico A, todas las colas del sistema son estables, independientemente de la capacidad de la DU. A medida que aumentamos la tasa de entrada las colas comienzan a ser más inestables hasta que llegamos al tráfico de entrada más alto como es el tráfico C. Las colas que más sufren son las de la DU. A pesar de ello, se nota una mejoría a medida que aumentamos la capacidad nominal de la unidad distribuida, mostrando los mejores resultados en el caso 3 tal y como se observa en la Figura 4.14.



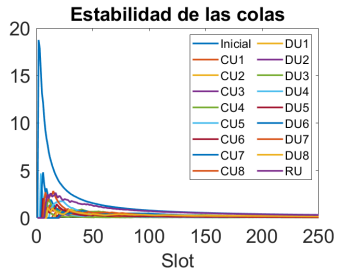
(a) Tráfico A (b) Tráfico B (c) Tráfico C

Figura 4.12: Estabilidad de las colas - Caso 1

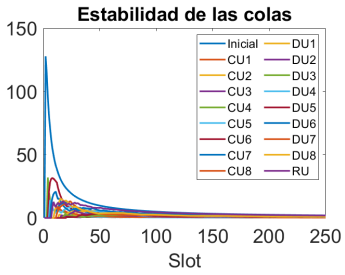


(a) Tráfico A (b) Tráfico B (c) Tráfico C

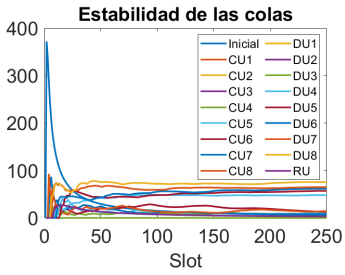
Figura 4.13: Estabilidad de las colas - Caso 2



(a) Tráfico A



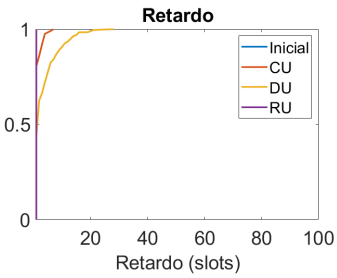
(b) Tráfico B



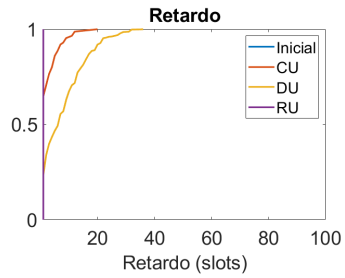
(c) Tráfico C

Figura 4.14: Estabilidad de las colas - Caso 3

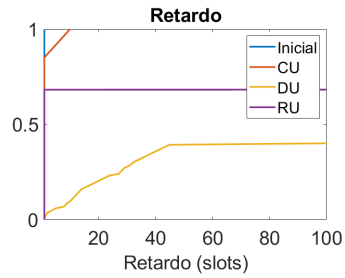
El retardo de los paquetes se muestra mediante la Función de Distribución Acumulativa Empírica mostrada en las Figuras 4.15, 4.16, 4.17. Por una parte, el retardo es mayor a medida que aumentamos la tasa de entrada dando lugar a resultados como los de la Figura 4.15c, donde más del 40% de los paquetes no pasan de la DU debido a que su capacidad es la menor de todas. Por otra parte, las diferencias en retardo entre los casos 2 y 3 es bastante pequeña, siendo incluso inferior para el Tráfico B, de lo que se puede deducir que el rendimiento en cuanto a retardo es similar para una DU al 50% de su capacidad que con una DU al 90% de su capacidad.



(a) Tráfico A

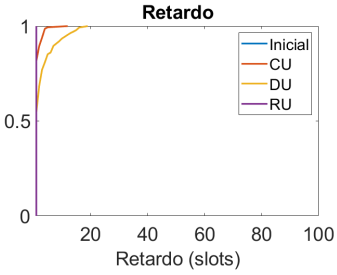


(b) Tráfico B

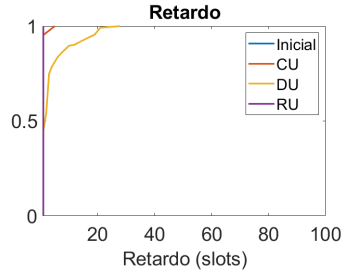


(c) Tráfico C

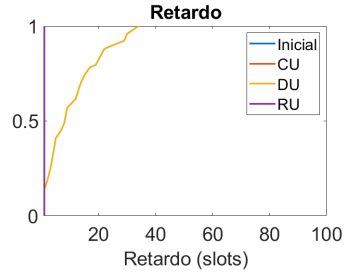
Figura 4.15: Retardo de los paquetes - Caso 1



(a) Tráfico A

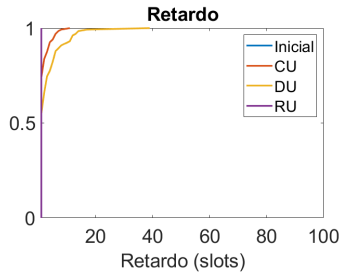


(b) Tráfico B

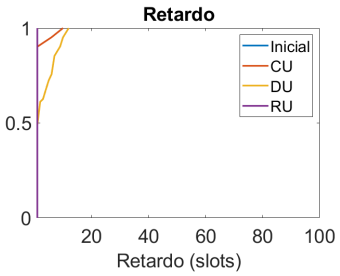


(c) Tráfico C

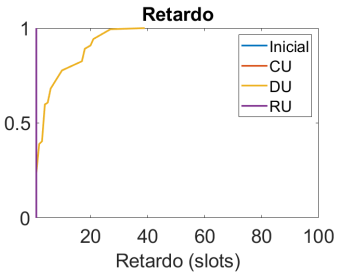
Figura 4.16: Retardo de los paquetes - Caso 2



(a) Tráfico A



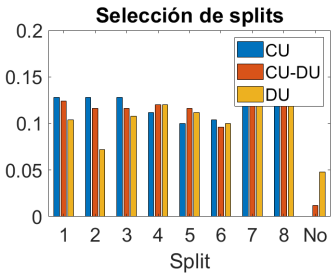
(b) Tráfico B



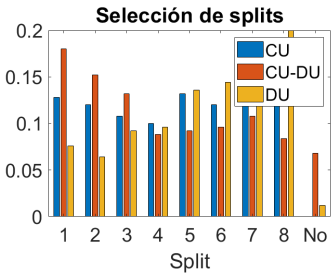
(c) Tráfico C

Figura 4.17: Retardo de los paquetes - Caso 3

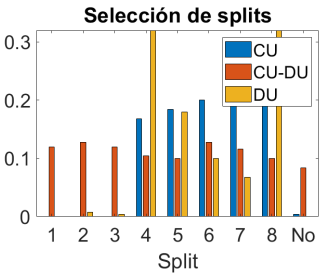
La selección de *splits* se puede ver en las Figuras 4.18 4.19 4.20. Los resultados son prácticamente iguales para el tráfico A y el tráfico B, pues en ningún caso superan las capacidades de enlace de la CU. La diferencia aparece para el tráfico C, para el que se selecionan las opciones 1, 2 y 3 con una probabilidad cercana a 0. El motivo es que la tasa de entrada es mayor que las capacidades de los enlaces asociadas a estas opciones. Este hecho demuestra que *Backpressure* se adapta a las variaciones de tráfico del sistema, tendiendo a una mayor centralización en la CU a medida que aumenta el tráfico en el sistema.



(a) Tráfico A

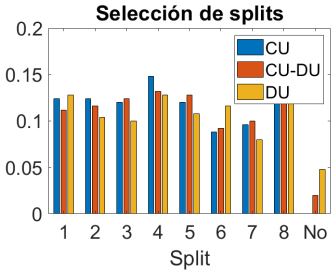


(b) Tráfico B

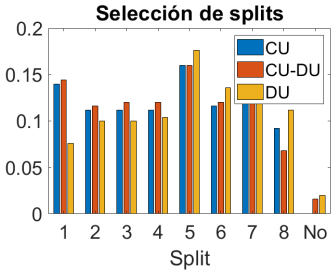


(c) Tráfico C

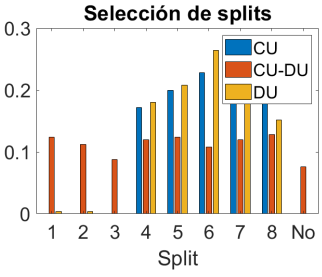
Figura 4.18: Selección de *splits* - Caso 1



(a) Tráfico A

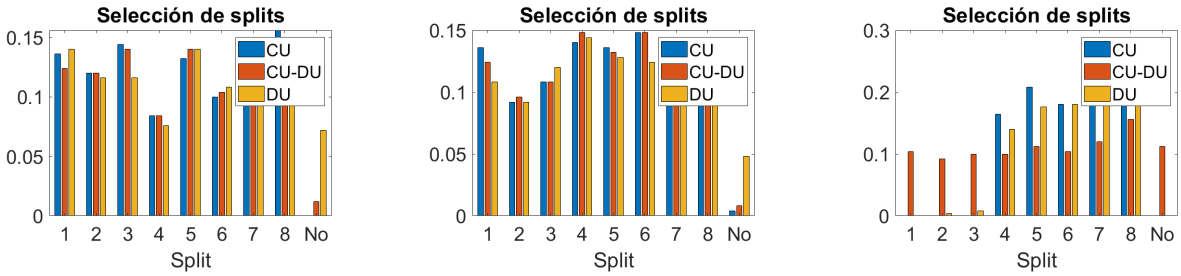


(b) Tráfico B



(c) Tráfico C

Figura 4.19: Selección de *splits* - Caso 2



(a) Tráfico A (b) Tráfico B (c) Tráfico C

Figura 4.20: Selección de *splits* - Caso 3

4.3 Escenario 2

El tercer y último escenario que se plantea es una extensión del **escenario 1**, en el que se amplía el número de estaciones base de una a tres. Al existir más de una estación base, los recursos de la *Central Unit* son compartidos, puesto que se encuentran centralizados. El algoritmo *Backpressure* ahora debe buscar la selección óptima de *splits* teniendo en cuenta todas las estaciones base.

Para dotar de más realismo a las simulaciones, se introduce una limitación en la capacidad nominal de todas las CUs como conjunto, de manera que no todas las combinaciones de *splits* sean posibles para el segmento de red de la CU. El escenario con limitaciones de capacidad computacional en la CU se estudia en el caso 2, mientras que en el caso 1 de este apartado se estudia el comportamiento del algoritmo sin limitaciones.

Para las tres estaciones base se utiliza la misma configuración de capacidades para la CU y la DU, en la que la DU tiene un 25% de la capacidad de la CU, tal y como se muestra en la Tabla 4.4.

Tabla 4.4: Capacidades de los enlaces y tasa de entrada para el Escenario 2

Capacidad enlaces (`pkts/s`)	CU	CU-DU	DU	RU
C-RAN	639.23	6000.00	33400.00	40000.00
Intra-PHY	835.00	6000.00	668.00	40000.00
MAC/PHY	1248.60	6000.00	324.27	40000.00
Intra-MAC	1535.63	6000.00	271.54	40000.00
RLC/MAC	1994.03	6000.00	233.57	40000.00
Intra-RLC	2343.86	6000.00	218.30	40000.00
PDCP/RLC	2842.55	6000.00	204.91	40000.00
RRC/PDCP	4948.15	6000.00	182.51	40000.00
CU Cluster	20 GOPS			
DU	5 GOPS			

4.3.1 Caso 1: Sin limitación en la capacidad conjunta de la CU

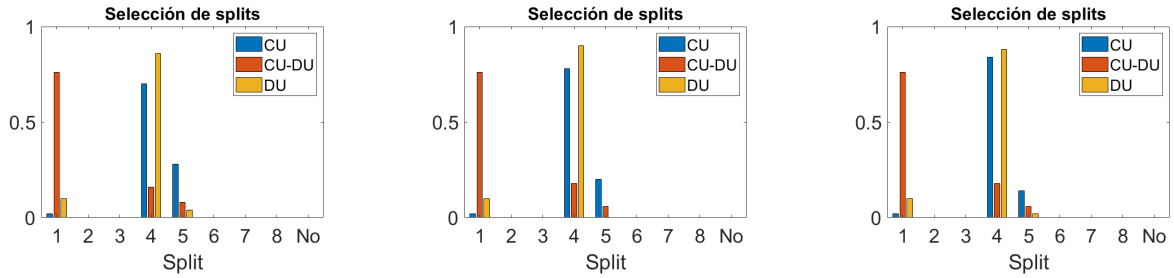
El primero de los casos planteados tiene como objetivo observar el comportamiento de *Backpressure* en la selección de *splits* sin que la capacidad nominal de los recursos compartidos en la CU sea un factor limitante, para posteriormente compararlo con otros casos en los que sí existe esta limitación.

Se realizarán dos simulaciones con dos tasas de entrada diferentes, el tráfico A con $\lambda = 900$ paquetes por *slot* y el tráfico B con $\lambda = 2500$ paquetes por *slot*.

Los resultados de estabilidad y retardo se incluyen en los Anexos B y C. Las colas de la DU son inestables puesto que la tasa de entrada tiene una media más alta que la mayoría de las capacidades de sus enlaces lo que conlleva también asociado un mayor retardo. El comportamiento para las tres estaciones base es muy similar.

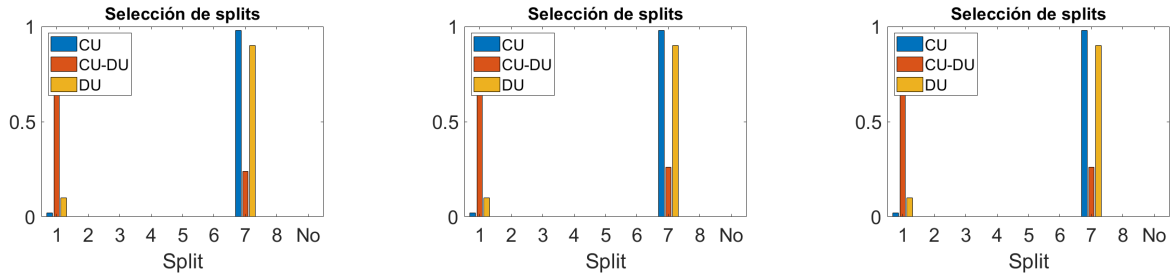
Los resultados más interesantes son los relativos a la selección de *split* que se muestran en las Figuras 4.21 4.22. En la primera figura se puede observar cómo se seleccionan con mayor probabilidad los *splits* 4 y 5 mientras que en la segunda figura el *split* 7 es el más seleccionado.

Esto demuestra que *Backpressure* se adapta al tráfico del sistema y optimiza los recursos existentes ya que, por ejemplo, no recurre a una centralización total. También cabe destacar cómo en el segundo segmento de la red (enlace CU-DU) se selecciona en todos los casos la opción 1. En este caso, esta opción equivale en la práctica a la opción de 'No transmitir', puesto que la mayoría de las veces que se selecciona esta opción no hay paquetes en las colas asociadas a ella, ya que se encuentran en las colas asociadas a las opciones 4 y 5 en el primer caso y a la opción 7 en el segundo. Esto ocurre porque a la hora de escoger la combinación óptima, *Backpressure* da el mismo peso a la opción 1 y a la de 'No transmitir' ya que su función es la misma.



(a) Estación 1 (b) Estación 2 (c) Estación 3

Figura 4.21: Selección de *splits* - Caso 1, Tráfico A



(a) Estación 1 (b) Estación 2 (c) Estación 3

Figura 4.22: Selección de *splits* - Caso 1, Tráfico B

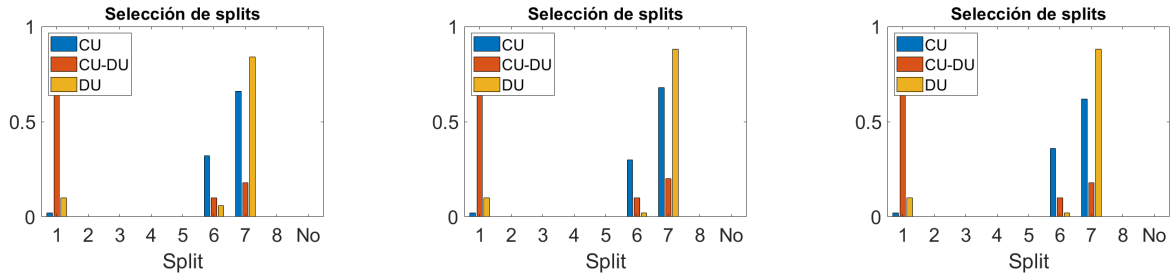
4.3.2 Caso 2: Limitación en la capacidad conjunta de la CU

El segundo de los casos consiste en limitar la capacidad total compartida de la unidad centralizada, de manera que no sea posible escoger todas las combinaciones de selección de *split*.

Para ello se tomarán como punto de partida los parámetros del Caso 1 B, en el que la combinación más frecuente en la CU era [7 7 7], para lo que se necesita una capacidad de cómputo de 8527.65 paquetes por *slot*. En una primera simulación se limitará la capacidad solo un tanto por debajo de la anterior, concretamente a 8500 paquetes por *slot*, de manera que la combinación óptima ya no es posible. Después, se limitirá de manera más drástica la capacidad total a 6000 paquetes por *slot* para comprobar cuál es la tendencia que sigue el *scheduler* en la selección de *splits*.

Los resultados de estabilidad y retardo se incluyen en los Anexos B y C. Son inestables las colas de la DU puesto que la tasa de entrada tiene una media más alta que la mayoría de las capacidades de sus enlaces. Esto conlleva también un retardo asociado a las unidades distribuidas de las estaciones base. El comportamiento para las tres estaciones base es muy similar.

Nuevamente los resultados más interesantes para la primera limitación de capacidad son los relativos a la selección de *split* que se muestran en la Figura 4.23.

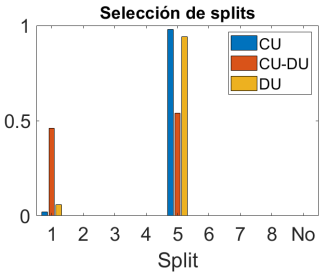


(a) Estación 1 (b) Estación 2 (c) Estación 3

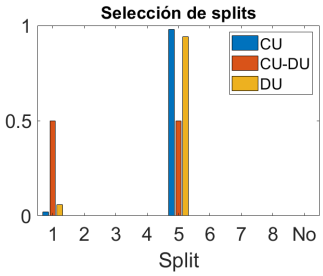
Figura 4.23: Selección de *splits* - Caso 2, Capacidad Total = 8500 paquetes por *slot*

Al limitar la capacidad total, la combinación [7 7 7] deja de ser posible por lo que *Backpressure* selecciona los *splits* más ajustados a la combinación original, como son las combinaciones [6 7 7], [7 6 7] o [7 7 6]. Los resultados demuestran cómo *Backpressure* se sigue adaptando al tráfico de entrada y a las capacidades del sistema, tratando por igual a las tres estaciones base y tratando al sistema como conjunto.

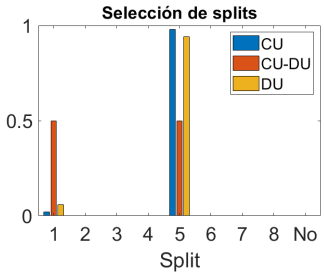
Después se ha procedido a limitar la capacidad total de manera más drástica para observar si la selección de *splits* sigue la misma tendencia. En este caso, la capacidad conjunta de la CU se rebaja a 6000 paquetes por *slot*. Los resultados en la Figura 4.24 muestran cómo se selecciona para todas las estaciones la misma combinación en la CU [5 5 5] cuya capacidad de procesado requerida es de 5983, lo que refuerza la conclusión de que *Backpressure* escoge la opción más ajustada a la capacidad total de la CU tratando de perjudicar lo menos posible la estabilidad de las colas y el retardo de los paquetes.



(a) Estación 1



(b) Estación 2



(c) Estación 3

Figura 4.24: Selección de *splits* - Caso 2, Capacidad Total = 6000 paquetes por *slot*

Chapter 5

Conclusiones

En este trabajo se ha introducido el modelo de arquitectura 5G vRAN y el concepto de la división funcional flexible. Además, se ha descrito un algoritmo para la selección dinámica de *splits*, capaz de tomar decisiones en tiempo real de simulación. El algoritmo *Backpressure*, basado matemáticamente en teoría de Lyapunov [12] que asegura la estabilidad de sistemas de colas. Además, en los escenarios estudiados se ha visto que este algoritmo tiene la habilidad de equilibrar las cargas de tráfico, mejorar el *throughput* y es simple de implementar en redes de comunicaciones.

Se ha comparado *Backpressure* con otros dos algoritmos que no ofrecen la selección dinámica de *splits* en tiempo real como son *Round Robin* y *Weighted Fair Queueing* y los resultados han demostrado la superioridad del primero frente al resto en términos de retardo de los paquetes y estabilidad de las colas.

Después, se ha analizado el rendimiento de *Backpressure* en escenarios realistas, donde se han tomado diferentes tasas de entrada y se ha hecho un barrido en la capacidad de la *Distributed Unit (DU)*. Los resultados han indicado que *Backpressure* adapta la selección de *splits* a la tasa de tráfico de entrada, favoreciendo una utilización eficiente de los recursos computacionales de la *Central Unit (CU)*. Sin embargo, el retardo de los paquetes depende enormemente de la capacidad computacional de la DU.

Por último, se han implementado escenarios con más de una estación base en los que los recursos computacionales compartidos entre las CUs son un factor limitante en la selección de *splits*. Los resultados han demostrado que incluso en esta situación *Backpressure* escoge la combinación de *splits* que aprovecha de manera más eficiente los recursos computacionales, tratando a todas las estaciones base como un único sistema en el que ninguna estación se ve más perjudicada que otra.

El desarrollo y resultados obtenidos en este trabajo permiten acometer varias extensiones. En primer lugar se abordará el análisis de escenarios más complejos en número y heterogeneidad de estaciones base. Asimismo, el modelo de red basado en colas permitiría introducir *switches* presentes en la red de *fronthaul* de forma que el algoritmo pueda seleccionar el nivel de *split* a la vez que reconfigura el encaminamiento de la red.

Anexo A

Capacidades de los enlaces para el Escenario 1

Tabla A.1: Capacidades de los enlaces y tasa de entrada para el Caso 1

Capacidad enlaces (`pkts/s`)	CU	CU-DU	DU	RU
C-RAN	639.23	6000.00	13360.00	15000.00
Intra-PHY	835.00	6000.00	267.20	15000.00
MAC/PHY	1248.60	6000.00	129.71	15000.00
Intra-MAC	1535.63	6000.00	108.72	15000.00
RLC/MAC	1994.03	6000.00	93.43	40000.00
Intra-RLC	2343.86	6000.00	87.32	15000.00
PDCP/RLC	2842.55	6000.00	81.96	15000.00
RRC/PDCP	4948.15	6000.00	73.01	15000.00
CU Cluster	20 GOPS			
DU	2 GOPS			

Tabla A.2: Capacidades de los enlaces y tasa de entrada para el Caso 2

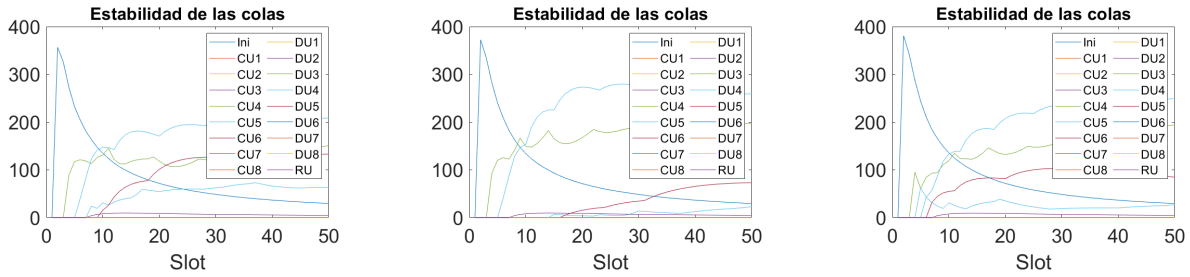
Capacidad enlaces (`pkts/s`)	CU	CU-DU	DU	RU
C-RAN	639.23	6000.00	66800.00	70000.00
Intra-PHY	835.00	6000.00	1336.00	40000.00
MAC/PHY	1248.60	6000.00	648.54	70000.00
Intra-MAC	1535.63	6000.00	543.09	40000.00
RLC/MAC	1994.03	6000.00	467.13	70000.00
Intra-RLC	2343.86	6000.00	436.60	70000.00
PDCP/RLC	2842.55	6000.00	409.82	70000.00
RRC/PDCP	4948.15	6000.00	365.03	70000.00
CU Cluster	20 GOPS			
DU	10 GOPS			

Tabla A.3: Capacidades de los enlaces y tasa de entrada para el Caso 3

Capacidad enlaces (`pkts/s`)	CU	CU-DU	DU	RU
C-RAN	639.23	6000.00	120240.00	150000.00
Intra-PHY	835.00	6000.00	2404.00	40000.00
MAC/PHY	1248.60	6000.00	1167.00	150000.00
Intra-MAC	1535.63	6000.00	977.00	40000.00
RLC/MAC	1994.03	6000.00	840.00	150000.00
Intra-RLC	2343.86	6000.00	785.00	150000.00
PDCP/RLC	2842.55	6000.00	737.00	150000.00
RRC/PDCP	4948.15	6000.00	657.00	150000.00
CU Cluster	20 GOPS			
DU	18 GOPS			

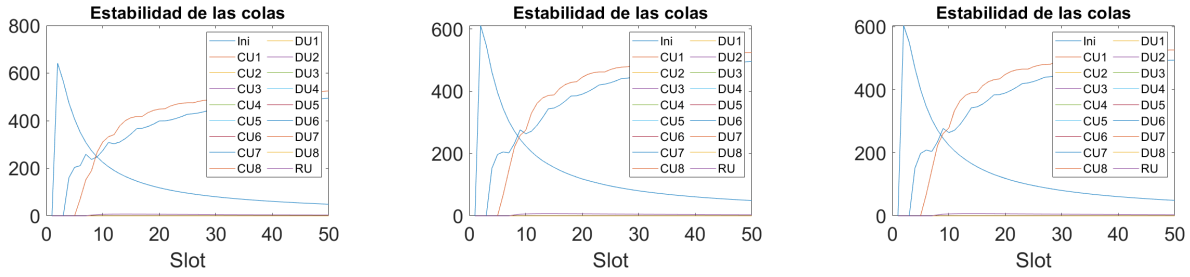
Anexo B

Estabilidades de las colas del Escenario 2



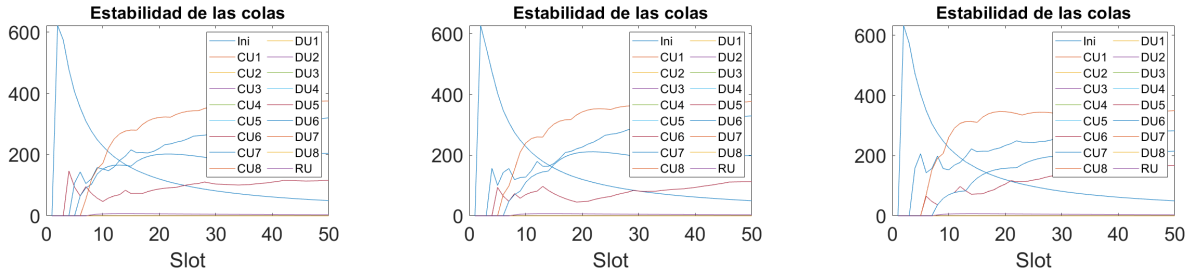
(a) Estación 1 (b) Estación 2 (c) Estación 3

Figura B.1: Estabilidad de las colas - Caso 1



(a) Estación 1 (b) Estación 2 (c) Estación 3

Figura B.2: Estabilidad de las colas - Caso 2

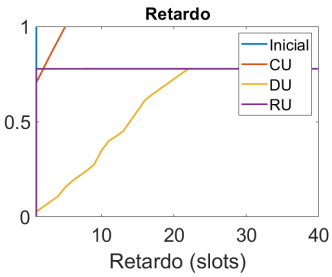


(a) Estación 1 (b) Estación 2 (c) Estación 3

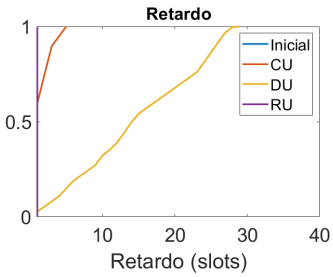
Figura B.3: Estabilidad de las colas - Caso 3

Anexo C

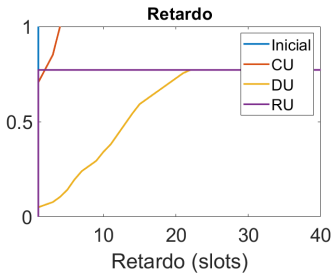
Retardo de los paquetes del Escenario 2



(a) Estación 1

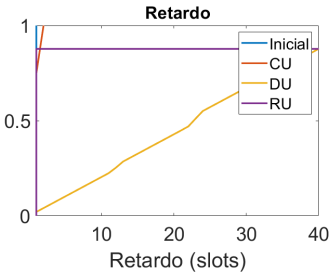


(b) Estación 2

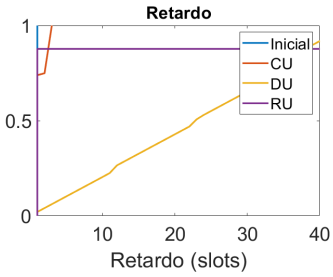


(c) Estación 3

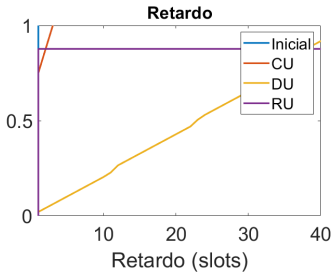
Figura C.1: Retardo - Caso 1



(a) Estación 1

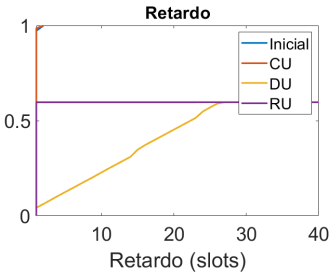


(b) Estación 2

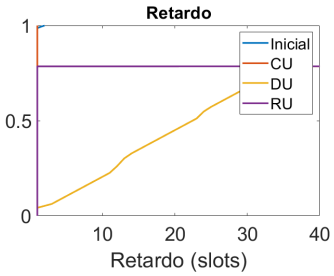


(c) Estación 3

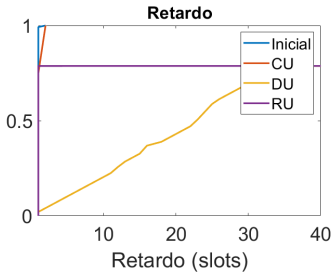
Figura C.2: Retardo - Caso 2



(a) Estación 1



(b) Estación 2



(c) Estación 3

Figura C.3: Retardo - Caso 3

Bibliografía

[1] Cisco. Cisco Annual Internet Report (2018-2023). White Paper. Technical report, Cisco, 2020.

[2] Unión Internacional de Telecomunicaciones. Sentando las bases para la 5G: Oportunidades y desafíos. Technical report, ITU-D, 2018.

[3] Luis Francisco Díez Fernández, Víctor González Carril, and Ramón Agüero Calvo. Optimización conjunta del nivel split y scheduling en redes 5G. XIV Jornadas de Ingeniería Telemática (JITEL), Zaragoza, 2019, 2019.

[4] NGMN Alliance. 5G White Paper. Technical report, NGMN Alliance, 2015.

[5] Line M. P. Larsen, Aleksandra Checko, and Henrik L. Christiansen. A Survey of the Functional Splits Proposed for 5G Mobile Crosshaul Networks. *IEEE Communications Surveys Tutorials*, 21(1):146–172, Firstquarter 2019.

[6] Davit Harutyunyan and Roberto Riggio. Flex5G: Flexible Functional Split in 5G Networks. *IEEE Transactions on Network and Service Management*, 15(3):961–975, Sep. 2018.

[7] Aleksandra Checko, Henrik L. Christiansen, Ying Yan, Lara Scolari, Georgios Kardaras, Michael S. Berger, and Lars Dittmann. Cloud RAN for Mobile Networks—A Technology Overview. *IEEE Communications Surveys Tutorials*, 17(1):405–426, Firstquarter 2015.

[8] Alberto Martínez Alba, Jorge Humberto Gómez Velásquez, and Wolfgang Kellerer. An adaptive functional split in 5G networks. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 410–416, April 2019.

[9] 3rd Generation Partnership Project. Study on New Radio Access Technology: Radio Access Architecture and Interfaces. Technical report, 3GPP, 2017.

[10] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1948, Dec 1992.

[11] Zhenzhen Jiao, Baoxian Zhang, Cheng Li, and Hussein T. Mouftah. Backpressure-based routing and scheduling protocols for wireless multihop networks: A survey. *IEEE Wireless Communications*, 23(1):102–110, February 2016.

[12] Michael Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan Claypool, 2010.

[13] Bjorn Debaillie, Claude Desset, and Filip Louagie. A Flexible and Future-Proof Power Model for Cellular Base Stations. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pages 1–7, May 2015.

[14] International Telecommunication Union. M. 2083: IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond. Technical report, ITU-R, 2015.